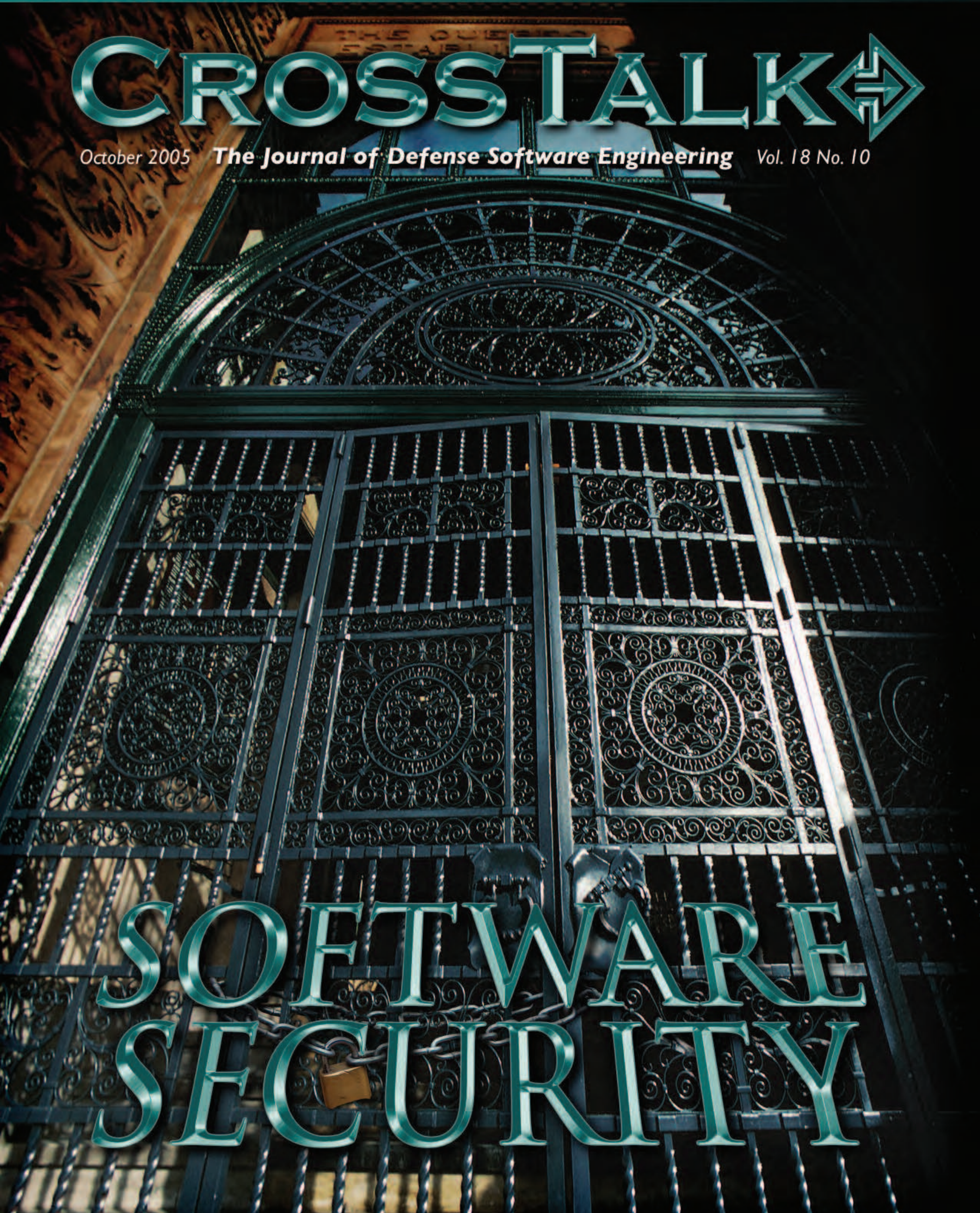
SPECIAL ISSUE: ENSURING SECURE SOFTWARE

CROSSTALK

October 2005 **The Journal of Defense Software Engineering** Vol. 18 No. 10



Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **OCT 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **CrossTalk: The Journal of Defense Software Engineering. Volume 18, Number 10, October 2005**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **OO-ALC/MASE,6022 Fir Ave,Hill AFB,UT,84056-5820**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**32**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

On The Cover
Cover Design by Kent Bingham.

Additional art services provided by Janna Jensen. jensendesigns@aol.com

CrossTalk

76 SMXG Co-Sponsor	*Kevin Stamey*
309 SMXG Co-Sponsor	*Randy Hill*
402 SMXG Co-Sponsor	*Tom Christian*
DHS Co-Sponsor	*Joe Jarzombek*
Publisher	*Tracy Stauder*
Associate Publisher	*Elizabeth Starrett*
Managing Editor	*Pamela Palmer*
Associate Editor	*Chelene Fortier-Lozancich*
Article Coordinator	*Nicole Kentta*
Phone	(801) 775-5555
Fax	(801) 777-8069
E-mail	crosstalk.staff@hill.af.mil
CrossTalk Online	www.stsc.hill.af.mil/crosstalk

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the United States Air Force (USAF) and the United States Department of Homeland Security (DHS). USAF co-sponsors: Oklahoma City-Air Logistics Center (ALC) 76 Software Maintenance Group (SMXG), Ogden-ALC 309 SMXG, and Warner Robins-ALC 402 SMXG. DHS co-sponsor: National Cyber Security Division of the Office of Infrastructure Protection.

The USAF Software Technology Support Center (STSC) is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal. CrossTalk's mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

S T S C

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 24.

309 SMXG/MXDB
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with CrossTalk.

Trademarks and Endorsements: This Department of Defense (DoD) journal is an authorized publication for members of the DoD. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, or the STSC. All product names referenced in this issue are trademarks of their companies.

Coming Events: Please submit conferences, seminars, symposiums, etc. that are of interest to our readers at least 90 days before registration. Mail or e-mail announcements to us.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-0857 or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Software Security: Shifting the Paradigm From Patch Management To Software Assurance

Dependency on information technology places software assurance as a key element of national security and homeland security. Vulnerable software is a risk to a broad spectrum of business and mission operations, including everything from process control systems to commercial application products that support and enable them. Software enables and controls the nation's critical infrastructure, and to ensure the integrity of that infrastructure, the software must be reliable and secure. However, informed consumers have growing concerns about software security, suppliers' capabilities to exercise a minimum level of responsible practice, and the scarcity of practitioners with requisite competencies to build secure software. Security-enhanced processes and technologies are required to build trust into software acquired and used by government and those who operate our nation's critical infrastructure.

The Department of Homeland Security (DHS) Software Assurance Program is grounded in the "National Strategy to Secure Cyberspace." DHS began the Software Assurance Program as a focal point to partner with the private sector, academia, and other government agencies to improve software development and acquisition processes. Through public-private partnerships, the Software Assurance Program framework shapes a comprehensive strategy that addresses people, process, technology, and acquisition throughout the software life cycle. Our efforts seek to shift the paradigm away from patch management and to achieve a broader ability to routinely develop and deploy trustworthy software products. These efforts will contribute to the production of higher quality, more secure software. The DHS Software Assurance Program is designed to lead the development of practical guidance, review tools, and promote research and development investment in cyber security. The overall goal is secure and reliable software supporting mission requirements, enabling more resilient operations.

Through hosting and co-hosting various forums and workshops, we will continue to leverage collaborative efforts of public-private working groups. DHS initiatives such as the "Build Security In" Web site and the "Software Assurance Common Body of Knowledge" will continue to evolve and provide practical guidance to software developers, architects, and educators on how to improve the quality, reliability, and security of software. These two initiatives are discussed this month in *Engineering Security Into the Software Development Life Cycle* (see page 4) and *Creating a Software Assurance Body of Knowledge* (see page 5).

DHS collaboration with standards organizations is focused on evolving standards to reflect guidance for appropriate levels of responsible practice for software security. To be relevant in today's global economy that relies on outsourcing of software and information technology services, models and standards that provide criteria to guide and appraise process improvement and support international commerce must explicitly address security (see *Sixteen Standards-Based Practices for Safety and Security* on page 11).

This DHS-sponsored issue of CROSSTALK addresses the value of software security; I hope you will take the time to read, understand, and apply the principles and techniques discussed in this month's articles. I encourage you to discover more about our DHS Software Assurance Program and learn more about proven security practices by visiting us at <http://BuildSecurityIn.us-cert.gov> and join others in our expanding software assurance community of practice.

Joe Jarzombek

Joe Jarzombek, Project Management Professional (USAF Lt. Col., Retired)
Director for Software Assurance
National Cyber Security Division
Department of Homeland Security

Engineering Security Into the Software Development Life Cycle

Gary M. McGraw
Cigital Inc.

Nancy R. Mead
Software Engineering Institute

The Build Security In (BSI) initiative seeks to alter the way software is developed so that it is less vulnerable to attack when security is built in from the start. BSI is part of the Software Assurance Program within the Strategic Initiatives Branch of the National Cyber Security Division of the Department of Homeland Security. As part of the initiative, a BSI content catalog is available on the U.S. Computer Emergency Readiness Team Web site. It is intended for use by software developers and software development organizations who want information and practical guidance on how to produce secure and reliable software.

Today's large-scale, highly distributed, networked systems improve the efficiency and effectiveness of organizations by permitting whole new levels of organizational integration. However, such integration is accompanied by elevated risks of intrusion and compromise. Incorporating security and survivability capabilities into an organization's systems can mitigate these risks.

Typical software development life-cycle models do not focus on creating secure systems, and fall short when the goal is to develop systems with a high degree of assurance [1]. If addressed at all, security issues are often relegated to a separate thread of project activity, with the result that security is treated as an add-on property. This isolation of security considerations from primary system-development tasks results in an unfortunate separation of concerns. Security should be integrated and treated on par with other system properties to develop systems with required functionality and performance that can also withstand failures and compromises [2].

The Build Security In Software Assurance Initiative

The Build Security In (BSI) Software Assurance Initiative seeks to alter the way that software is developed so that it is less vulnerable to attack and security is *built* in from the start. BSI is a project of the Strategic Initiatives Branch of the National Cyber Security Division (NCSD) of the Department of Homeland Security (DHS).

Figure 1: *BSI Content Catalog Components*

BEST PRACTICES
- Risk Management
- Project Management
- Assembly, Integration, and Evolution
- Architectural Risk Analysis
- Threat Modeling
- Measurement Code Analysis
- Security Testing
- Measurement
- Incident Handling and Monitoring
- Penetration Testing
- White Box Testing
- Deployment and Operations

KNOWLEDGE
- Software Development Life Cycle
- Business Relevance
- Attack Patterns
- Principles
- Guidelines
- Historical Risks
- Coding Practices
- Coding Rules

TOOLS
- Modeling Tools
- Code Analysis
- Black Box Testing Tools

The initiative includes a BSI content catalog available on the U.S. Computer Emergency Readiness Team Web site at <http://BuildSecurityIn.us-cert.gov>. It is intended for use by software developers and software development organizations who want information and practical guidance on how to produce secure and reliable software. The catalog is based on the principle that software security is fundamentally a software engineering problem and must be addressed in a systematic way throughout the software development life cycle. The catalog either contains or links to a broad range of information about best practices, tools, guidelines, rules, principles, and other knowledge to help organizations build secure, reliable software.

Figure 1 identifies aspects of software assurance that are covered in the catalog and how the material has been organized. It categorizes catalog content according to best practices, knowledge, and tools, and includes business cases.

Best Practices

A significant portion of the BSI effort will be devoted to best practices that can provide the biggest return considering current best thinking, available technology, and industry practice.

Knowledge

Recurring patterns of software defects leading to vulnerabilities have been identified, and the BSI team is documenting detailed instructions on how to produce software without these defects under the headings "Guidelines," "Coding Practices," and "Coding Rules."

Tools

The BSI site includes information about the kinds of tools that can be used by both developers and security analysts to either detect or remove common vulnerabilities.

Business Cases

Business cases help convince industry to adopt secure software development best practices and to educate consumers on the need for software assurance. Each documented best practice addresses the business case for use of that practice. An overall business case framework will be included.

Future Plans

The DHS NCSD has invited representatives from industry, academia, and government to become involved. Part of this outreach activity includes seminars at which invited organizations can receive and share information about software assurance resources and help the stakeholder community understand both the need for building security in and the value of the Web site for providing relevant guidance. Content will be linked with reference sources and other materials made available through the DHS NCSD Software Assurance Program such as information in "security-enhancing the application software development life cycle" and the software assurance common body of knowledge, which provides a framework for education and training curriculum development in software assurance.◆

References

1. Marmor-Squires, A.B., and P.A. Rougeau. Issues in Process Models and Integrated Environments for Trusted Systems Development. Proc. of the 11th National Computer Security Conference. Fort George G. Meade, MD, Oct. 17-20, 1988: 109-113.
2. Mead, N.R., et al. "Managing Software Development for Survivable Systems." Annals of Software Engineering 2 (2001): 45-78.

Point of Contact

Jan Philpot
Software Engineering Institute
4301 Wilson BLVD STE 200
Arlington, VA 22203
Phone: (703) 908-8208
Fax: (703) 908-9317
E-mail: philpot@sei.cmu.edu

Creating a Software Assurance Body of Knowledge

Samuel T. Redwine Jr.
James Madison University

The Software Assurance Workforce Education and Training Working Group, composed of government, industry, and academic members, is currently taking a first step toward achieving adequate U.S. education and training on software assurance. It is defining the additional body of knowledge needed to acquire, develop, and sustain secure software beyond that normally required to produce and assure software where safety and security are not concerns.

In 2003, the Department of Defense (DoD) launched a software assurance initiative. In 2004, the Department of Homeland Security (DHS) joined in collaboration with DoD and other agencies and established its own Software Assurance Program[1], and DoD and DHS began to jointly sponsor semiannual software assurance forums.

While the term *software assurance* potentially encompasses assuring any property or functionality of software, the initiative encompasses safety and security and integrates practices from a number of disciplines (see Figure 1). Initially, the effort has concentrated on achieving and assuring security properties and functionality. This includes not only activities during development, but also the acquisition and sustainment processes.

This is driven by a growing demand for low-defect, secure software for crucial roles in defense and commerce often performed by commercial off-the-shelf products. However, current commonplace software specification, design, implementation, and testing practices provide users with software containing numerous defects and security vulnerabilities. Hence, the initiatives' Workforce Education and Training Working Group is currently addressing achievement of adequate U.S. education and training on software security, including training within government and industry, and curriculum needs within universities, colleges, and trade schools.

Defining Software Assurance Common Body of Knowledge

After deliberation, the working group decided to first create a description of the additional needed knowledge – beyond that required for *normal* software – to acquire, develop, and sustain secure software, including assurance of its security properties and functionality. The working group first identified the activities or aspects of activities relevant to secure software – beyond normal activities – and then asked, "What knowledge is needed to perform these activities?" Three difficult sub-questions exist:

1. What are the normal activities and their normal aspects?
2. What are the additional activities or aspects of activities that are relevant?
3. What knowledge is needed to perform these added activities?

Initially, the subgroup addressing software development has taken the Software Engineering Body of Knowledge Guide [1] as a working description of what is the normal knowledge.

The efforts to answer the second question benefit from a number of prior efforts, including the following:

- National Cyber Security Partnership Task Force report on "Processes to Produce Secure Software" [2].
- Safety and Security Extensions for Integrated Capability Maturity Models [3].
- National Institute of Standards and Technology Information System Security Project.

The working group also benefits from the expertise of its members and the work of other working groups and reviewers.

Some key knowledge not widely known even though associated with normal activities may be included. The intent is to ensure adequate coverage of requisite knowledge areas to enable professionals playing a number of roles in software engineering, systems engineering, and program management to identify knowledge and acquire competencies associated with software assurance. Because of this wide coverage and applicability, the intended product is officially called the "Software Assurance Common Body of Knowledge."

After several rounds of internal and external review, the initial report should include an introduction followed by four parts describing and identifying references for the additional knowledge required:

1. Common concepts and principles required across acquiring, developing, and sustaining secure software.
2. Development.
3. Post-Release Sustainment.
4. Acquisition and Supply.

The Software Assurance Common Body of Knowledge, initially released Oct. 3 at the DHS-DoD co-sponsored Software Assurance Forum, will be updated after public review and published in December 2005.◆

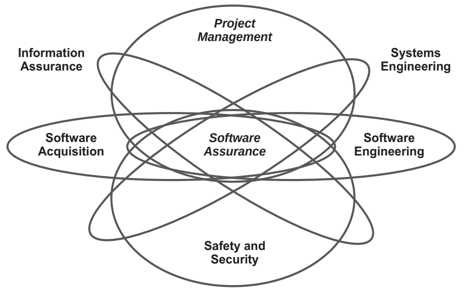


Figure 1: *Disciplines Contributing to Software Assurance*

References

1. Institute of Electrical and Electronics Engineers. Guide to the Software Engineering Body of Knowledge. Eds. P. Bourque and R. Dupuis. 2004 ed. Los Alamitos, CA: IEEE, 16 Feb. 2004.
2. National Cyber Security Partnership. Processes for Producing Secure Software: Towards Secure Software. Vols. I and II. Eds. S.T. Redwine Jr. and N. Davis. Washington: National Cyber Security Partnership, 2004.
3. Ibrahim, Linda, et al. Safety and Security Extensions for Integrated Capability Maturity Models. Washington: Federal Aviation Administration, Sept. 2004 <www.faa.gov/ipg>.

Note

1. See <http://BuildSecurityIn.us-cert.gov> for information about the DHS Software Assurance Program and related products.

Author Contact

Samuel T. Redwine Jr.
James Madison University
Computer Science MSC 4103
Harrisonburg, VA 22807
Phone: (540) 568-6305
E-mail: redwinst@jmu.edu

Designing for Disaster: Building Survivable Information Systems

Ronda R. Henning
Harris Corporation

Disaster recovery is a topic traditionally relegated to the operations staff as a mandatory function. Visions of natural disasters and terrorist acts keep information technology managers awake at night planning how to maintain normal business computing resources. But disasters can be as minimal as a broken water pipe, or an end-user who inadvertently unleashes an Internet worm within the enterprise. This article presents an alternative approach: designing survivability measures into an information system from the start. A discussion of survivability is presented, a methodology for survivable system design is defined, and an illustrative example is presented.

When a natural disaster strikes, a corporation normally places a disaster recovery plan into effect. These plans define how a corporate knowledge base is reconstituted after a catastrophic failure, allowing an enterprise to continue its daily functions. However, natural disasters are relatively rare occurrences. A corporation that leases space at a site hosting facility and purchases disruption insurance has allocated assets in advance, with potentially no return on those investments if a disaster does not occur [1]. In this regard, disaster recovery is like insurance.

With the ubiquity of the Internet, it has become more difficult to disrupt services for an extended period of time. Consumers expect 24-hour service or they take their Internet shopping elsewhere. Global enterprises now link what were isolated data centers to Enterprise Resource Planning systems to manage inventory and track consumer preferences. There is no downtime allowed in today's global economy.

Enter the concept of *survivable information systems*. Survivable information systems continue operating in various failure scenarios, although potentially in a degraded mode. Problems such as denial of service attacks, loss of a local network segment, or loss of a single data center are addressed by using various countermeasures or mechanisms to ensure continued system operation. When a traditional information system is subjected to failure conditions, it shuts down. By contrast, a survivable system continues functioning in support of the enterprise.

This article begins with a discussion of survivability, and contrasts survivability with the traditional disaster recovery and business continuity disciplines. A system survivability design methodology is presented that incorporates risk assessment and risk mitigation activities into the system development life cycle. Finally, representative examples of survivability mechanisms within the context of service-oriented system architectures are presented to assist those designing survivable information systems.

Survivability Defined

The Software Engineering Institute has conducted a comprehensive project on survivable information systems [2]. Survivability has been defined as "the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents." A mission is a set of high-level requirements that a system must fulfill to be considered successful. An organization may not have a mission statement as such, but every organization has some sort of vision that articulates the organization's ambitions.

Figure 1: *Survivability Versus Disaster Recovery Event Categories*

	Predictable	Unpredictable
Controllable	• Develop standard operating procedures	• Quick Response Team • Event Drills • Capture lessons learned
Uncontrollable	• Develop predictive models • Use indicators for avoidance	• Contingency planning

For example, if an electronic commerce server has an expectation of seven days a week, 24 hours a day availability, an unrecoverable disk crash that requires several hours to restore would be considered a failure to fulfill the mission. A *failure* is a potentially damaging event caused by a deficiency in the system or in an external element on which the system depends. An *accident* is defined as a randomly occurring and potentially damaging event such as a natural disaster that is thought of as externally generated. An *attack* is a potentially damaging event caused by an intelligent adversary.

What matters in the context of survivability is not so much the cause of a problem, but the system's response to that problem. To continue functioning, a system must respond to a failure, accident, or attack before the cause can be determined. That is, the system must react to the event, recover from it, and continue its mission. A classic example of a survivable system would be HAL, the all-knowing information system from "2001: A Space Odyssey" [3]. HAL reacted to attempted shut-down operations by protecting itself at the unfortunate expense of the Discovery's crew.

Hardy [4] categorizes events into four quadrants, see Figure 1. Survivable systems address the events that occur in all four quadrants. Events are categorized as either controllable or beyond the control of the system (uncontrollable), and predictable or unpredictable. Events that are predictable and controllable can be scheduled, or monitored and addressed before they become crisis. Unpredictable but controllable events can be addressed within the context of an incident response team such as those used to control malicious code attacks [5]. In a given situation, a survivable system reacts to all types of events and continues operation to fulfill its mission.

In Swanson [6], the notion of disaster

recovery is presented within the context of a major, catastrophic system failure that is caused by an external event and denies access to a normally used facility for an extended period of time. An example would be a hurricane or 100-year flood that decimates a building and the power, water, and transportation infrastructure required for support personnel. The goal of a disaster recovery plan is to allow resumption of operations as quickly as possible, often at an alternate site.

In Hardy's quadrant model, events that are unpredictable and uncontrollable cause activation of the disaster recovery plan. The concept of survivability incorporates disaster recovery, but extends the concept to include *all* events that may disrupt system operations.

Survivability Versus Continuity

Quirchmayr [7] and Nemzow [1] both distinguish the concept of continuity planning from the concept of disaster recovery. These authors consider continuity planning to involve the entire collection of personnel, processes, and procedures that, together with the computing assets, allow a business to continue functioning. In their respective models, the workflow associated with a system is considered at least as critical to an enterprise's survival as the actual hardware and software used to support the personnel.

In these models, action plans incorporate personnel considerations for extended-term outages. For example, if an enterprise's facility is without power, a continuity model would not only address auxiliary uninterruptible power supplies but also address the logistics of maintaining the fuel source, providing support to operations personnel, and providing alternate communications paths to other locations if necessary. Continuity planning integrates the entire business process model into the reaction and recovery tasks associated with enterprise information system operations.

The Goal of Survivability Planning

In an optimal situation, an enterprise can define an acceptable balance between the potential risk of a failure that would render a system unable to fulfill its mission and the cost associated with protective mechanisms. Bakry [8] proposes using economic analysis to optimize the cost of prevention versus risk of failure tradeoff, and the concept of a balanced solution is introduced.

This model is illustrated in Figure 2. For example, if an organization is extremely risk adverse, it can expend considerable assets on redundant computing environments and an alternate support staff. In contrast, an organization that believes a catastrophe will never happen to them might fulfill their plan with a package of writeable DVDs for media backup. In this instance, the organization accepts the risk of system loss in exchange for the cost savings associated with survivable safeguards

A Survivable Design Development Methodology

To improve organizational integration, information systems are becoming increasingly networked to trading partners, customers, and suppliers as well as other sites on the corporate network. In this model, *islands of automation* have been integrated into a network-enabled enterprise benefiting from their interconnectivity. Unfortunately, there is a downside to this model: A company may not have any knowledge about a virus attack that is spreading through a trading partner's network. In such an environment, a system architected with some degree of survivability such as firewalls, virus scanning, or intrusion detection/prevention appliances is most likely to fulfill its mission objectives.

How, then, does a system owner specify or design a system with survivability in mind? There are no generally accepted design methodologies in place to address the diverse events that can impact system survivability. Figure 3 illustrates a methodology that facilitates the integration of survivability characteristics within a traditional information system development framework.

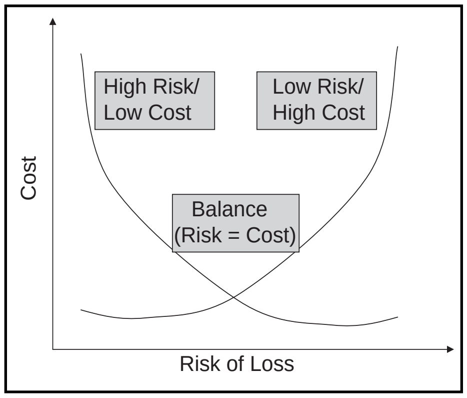


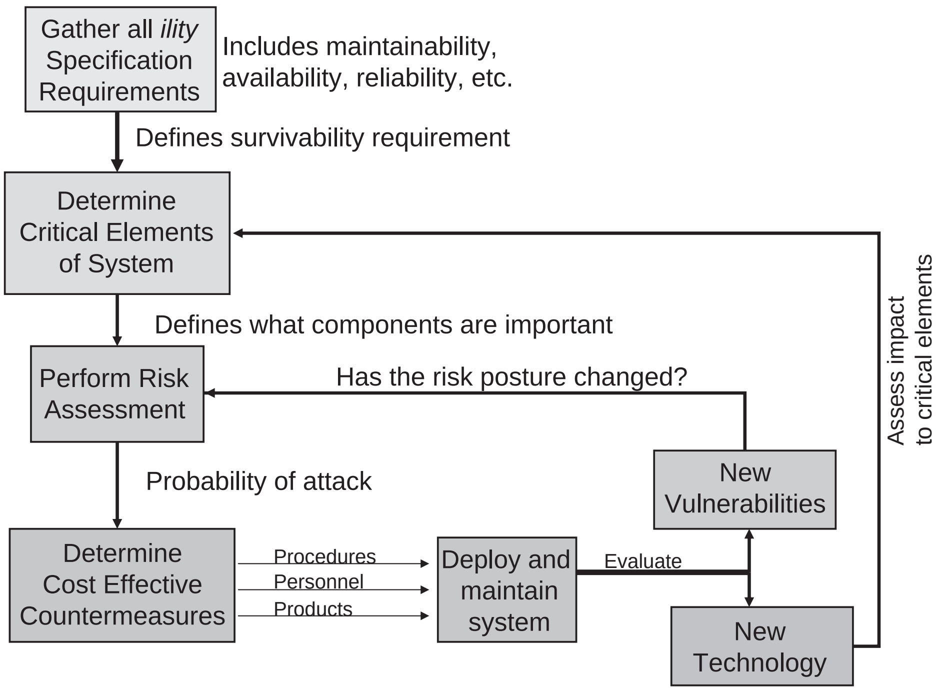
Figure 2: *Risk of Loss Versus Cost of Survivable Safeguard*

Gathering Survivability Related Requirements

To completely understand the survivability characteristics of an information system, it is essential to understand the system's requirements. In the language of system specification, survivability attributes are usually expressed in terms of specialty engineering disciplines and their requirements. These disciplines include availability, reliability, maintainability, and accountability as well as security and integrity – in Figure 3 they are collected as the *ility* requirements. Table 1 (see page 8) illustrates representative requirements in each of these disciplines that would impact the survivability characteristics of system architecture.

It is important to note that not all survivability constraints require technology-

Figure 3: *Survivability Integration Methodology*


Gather all *ility* Specification Requirements
Includes maintainability, availability, reliability, etc.
Defines survivability requirement
Determine Critical Elements of System
Defines what components are important
Perform Risk Assessment
Has the risk posture changed?
Probability of attack
New Vulnerabilities
Assess impact to critical elements
Determine Cost Effective Countermeasures
Procedures
Personnel
Products
Deploy and maintain system
Evaluate
New Technology

Requirement Family	Typical requirement statement	Potential survivability impact
Availability	The system shall be available for processing .9995 percent of the time.	Redundant hardware or dedicated communications paths.
Reliability	The system shall have less than 1 hour total downtime per year.	Hardware selection, or need for distributed architecture.
Maintainability	The system components shall be field-replaceable.	On-site replacement parts.
Integrity	The system shall protect information in transit from possible modification.	Secure hash or cryptographic sealing techniques.
Security	The system shall protect information at rest, during processing, and in transit.	Virtual private network (VPN) technology, disk encryption.

Table 1: *Requirement Families That Impact System Survivability*

based solutions. For example, racks containing computer hardware may be sealed with colored tape. If the seal is broken, maintenance personnel would inventory the components and, if necessary, run system diagnostics to detect potential system modifications. Similarly, if unauthorized access to a facility is a concern, a physical security policy that visitors must be escorted at all times provides an acceptable solution.

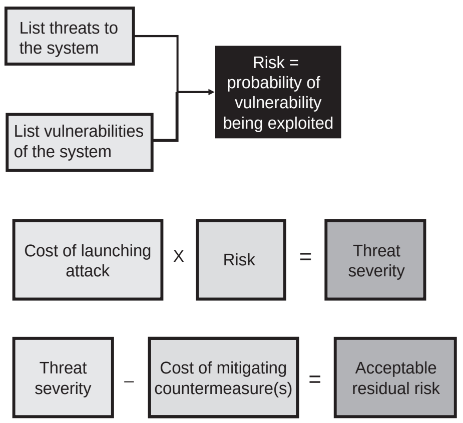
Determine Critical System Elements

From the survivability analysis, critical system elements should be identifiable. Criticality of elements may be based on connectivity requirements, processing capacity, or amount of data accessed. The objective of this process is to determine those elements of the system design whose failure or compromise would have the greatest impact on the operational system. From this decomposition, it is possible to determine which system components will require added care going forward in the development process.

Perform Risk Analysis

At this point, a risk analysis is required. This is not the traditional risk analysis that addresses risk to cost and schedule, but is a risk analysis that assesses the potential impact to the survivability posture of the system. Once the critical elements have been identified, the potential impact of adverse events must be evaluated. Figure 4, adopted from Panko's discussion [9], illustrates the traditional risk assessment process. In risk assessment, the potential threats to the system are enumerated. These threats are then evaluated in the context of system vulnerabilities. That is, a threat to a critical system element is only a threat if the opportunity to exploit a given vulnerability or group of vulnerabilities is present. Whitson [10] presents a basic overview of risk assessment.

Figure 4: *Traditional Risk Assessment Process*


List threats to the system
List vulnerabilities of the system
Risk = probability of vulnerability being exploited
Cost of launching attack
x
Risk
=
Threat severity
Threat severity
–
Cost of mitigating countermeasure(s)
=
Acceptable residual risk

Beyond determining the vulnerabilities, there is the cost associated with exploitation. For example, a risk of data tampering when information is only valid for less than a minute may carry a prohibitively high cost of exploitation. The cost of launching an attack coupled with the risk determines the threat severity. If information is updated every minute, an attacker would have to maintain an alternate data set of sufficient size to hide his intent until the attack is over. In such a case, the perceived cost of the attack would be relatively high, and the risk of detecting fraudulent data would also be relatively high. In such an instant, the threat severity, or consequences, if an attack was launched would be high for the simple fact that a deliberate, concerted attacker would be involved. When threat severity is evaluated in the context of countermeasures, the residual risk associated with system use is derived.

For example, a system that connects to the Internet may have a relatively high risk and a high threat severity. However, using a packet filtering firewall, a minimal set of network services required for the application, and *hardened* or security-conscious host configurations mitigate a considerable amount of risk. Risk to such a system could be further mitigated by using antivirus software and/or intrusion prevention technology.

The relative costs of architectural elements are significant inputs to the risk assessment process and impact the risk calculation. It should also be noted that some countermeasures may not be intuitively obvious. Creativity and innovation sometimes result in effective solutions for a given enterprise environment.

Countermeasures should also be cost-effective. Not all countermeasures are electronic and computer-intensive. Countermeasures can include standard operating procedures and policies. For example, if unauthorized facility access is a high-risk item, a cost-effective countermeasure could be limiting computer room access to authorized personnel by applying access limiting devices (i.e., locks with keys or smart cards). A guard dog turned loose at night can be just as effective as a sophisticated electronic alarm system.

Deploy and Maintain System

Once the countermeasures have been identified and deployed, the system must be maintained in a survivable state. For example, a system that depends upon anti-virus software must have the latest malicious code signature files downloaded when available. Application updates, or patches, must be tested for compatibility with the application environment and deployed across the enterprise. The best countermeasures in the world do not work if they are not maintained and enforced.

As the system matures, it must be continuously evaluated. For example, all the vulnerabilities associated with a given commercial product may not be applicable to a specific application of the system. A system may not use a given network service, so a patch deployment can be deferred. A new, improved version of a commercial off-the-shelf software component may become available, providing additional functionality without custom software development. In both cases, the relative risks associated with updating the system's architecture must be weighed against the potential risks that could be introduced into the survivability posture.

Threats are continuously evolving as new vulnerabilities and exploits are identified. The risk assessment activity is an ongoing part of the system life cycle. The residual risk associated with the system's survivability posture must be updated on a regular basis to reflect the current system architecture and the current threat environment. A survivability assessment that does not reflect the current state of the system architecture is not a useful document and may inaccurately reflect the risk posture of the system.

Putting the Model to Work

The survivability model described above

has been applied to several architecture development efforts, reflecting diverse environments and their unique operational characteristics. This section discusses how the model was applied to three specific cases.

The first example is a high throughput transaction-processing application. This particular system development was a *renovation* of an existing system that, while still performing adequately, was rapidly becoming unreliable. The fact that a data warehouse hardware platform had reached its end of life made modernization imperative. On first investigation, a high throughput distributed client-server model might have served the purpose. Unfortunately, data replication across the environment could not be supported with the desired reliability (five minutes of downtime per year), and the system maintained highly confidential legal information (criminal records). When the reliability requirements were factored into the equation, a high capacity centralized data warehouse environment with internal transaction process monitoring was a more efficient architecture.

For the second example, a mission-critical networking infrastructure was under consolidation and modernization. The existing network evolved from a series of stovepipe requirements, with each project managing and ordering its own network services. While this approach had served the organization well in the past, it was no longer cost effective or survivable in today's telecommunications environment.

A prioritization of services was undertaken by the customer, moving the network to a reliability-, maintainability-, and availability-based service model. For example, network services that required redundant connectivity and minimal downtime were segregated from traditional administrative-based services that could adapt to a next-business-day restoration. The net result: the organization has been able to reduce the number of redundant communications paths between facilities, reduce costs, and improve management visibility into critical services. Spare components are pre-positioned at strategically placed depot installations instead of stored at every site location.

The third example is a network information resource, responsible for routing user requests to the most probable source of the requested information. In this application, a user's clearance level, bandwidth, and intended use of the information are factored into satisfying the user's query. The application in question has applied distributed client/server architecture used to localize the data storage to its most logical requesters. For example, Pacific Command analysts do not normally explore European data sets. Data has been distributed to the most likely user base with replicated backup services on other servers.

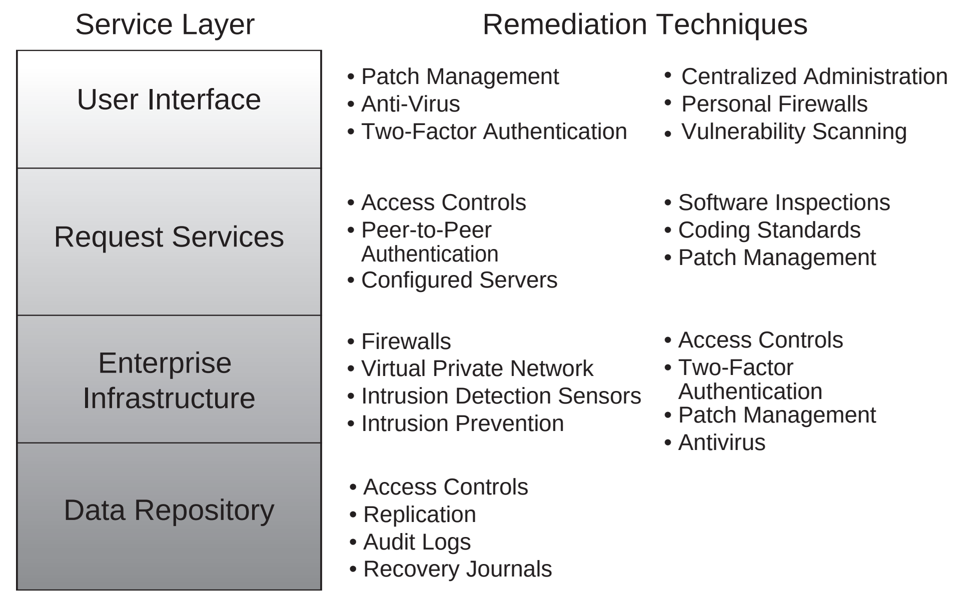


Figure 5: *Service-Oriented Architecture With Remediation Techniques*

"Once the countermeasures have been identified and deployed, the system must be maintained in a survivable state ... The best countermeasures in the world do not work if they are not maintained and enforced."

A Service-Oriented Example for the Future

Service-oriented architectures (SOAs) decouple data from both the user and the processing services in a layered structure. The most critical data in a SOA may be the processing required to fulfill a user request, or it may be the data repository that contains information vital to the enterprise's mission. Figure 5 illustrates a representative SOA with illustrative remediation techniques that could be applied at each layer.

In most enterprises, the user interfaces are assumed to execute on a standard enterprise desktop system (i.e., Linux, Windows, or Macintosh). Each enterprise configures and manages their desktops differently, depending on the information technology budget and capabilities of the end users. As a result, best practice desktop computing practices are usually applied such as virus scanning and/or patch management. Due to the multi-purpose nature of user desktops, they are considered high risk for introducing potential vulnerabilities into the enterprise. Backup media are usually the responsibility of the end user, unless the enterprise provides global backup services.

The request services layer may represent a significant investment for the enterprise. This layer usually addresses pre- and post-processing needed to make the data meaningful to the user's presentation environment. For example, an application that contains a corporate knowledge base or expert system may represent irreplaceable domain expertise. This type of application can be applied as an analyst's aid to filter large data sets. In these cases, survivability can be enhanced through using good software development and maintenance practices, including configuration management and code escrow.

The enterprise infrastructure is the lifeblood of a services-based architecture. Without the enterprise communication services, data cannot be moved across the layers of the processing architecture in response to user requests. Because the enterprise infrastructure is responsible for both the availability of the information and the integrity of the data in transit, it represents a significant risk to the survivability posture of the enterprise. This is the reason most modern enterprise infrastructures are well protected. Mechanisms employed at the infrastructure layer include virtual private networks, intrusion detection sensors, and firewalls. Each enterprise determines the residual risk associated with the infrastructure and defines appropriate countermeasures as required for the applications it supports.

The data repository may represent the most critical portion of the application. It could contain the corporation's financial records, business intelligence information, or customer records. This information, frequently gathered over extended periods of time, may be the most irreplaceable in an enterprise. As such, the data repository is usually subject to layered data integrity and defense mechanisms. These may include replicating updates to alternate geographically dispersed sites, using journaling and recovery to ensure transactions are saved to the database completely, and using additional authentication techniques for database restructuring. The repository contains the data that allows the applications to perform their analytical or reporting functions, and is vital to the extended life of the enterprise.

Conclusion

The countermeasures in the simple example described above are representative of the types of mechanisms that can be deployed to enhance the survivability of an information system. However, no countermeasure should be deployed without completion of a cost/benefit analysis. There are times when a very elegant, 100 percent effective countermeasure is not the best fit for an enterprise: when deployment would require major upgrades to other portions of the information technology environment, or substantively damage the functionality of existing applications. The objective is to make an enterprise information system survivable, not inaccessible.

Survivability can be attained by augmenting the traditional system development paradigms with a relatively small set of process augmentations. The goal is creation of a risk-oriented model of the system that allows the owner/creator to make sound decisions about design alternatives impacting the survivability of the system. When such models are in place, they can effectively enhance the survivability posture of the system. In such environments, expenditures on major disaster recovery plans can be greatly reduced because the system has integrated reaction and recovery capabilities.

Survivability as a design consideration yields a more effective return on investment than expenditures for redundant hardware, hot site backup, and major disaster preparedness measures. The incorporation of survivability addresses all potential disaster scenarios, not just the most catastrophic, resulting in a more prepared organization that can react effectively to multiple event scenarios, in effect creating a more adaptable and agile system.◆

® CERT is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Information Sources

For additional information on survivability, the Software Engineering Institute at <www.sei.cmu.edu> has ongoing projects on survivable design, and runs the U.S. Computer Emergency Response Team (U.S.-CERT®) Coordination Center <www.us-cert.gov>. For additional information on contingency planning and disaster recovery, the National Institute of Standards and Technology hosts a Computer Security Resource Clearinghouse at <www.csrc.nist.gov>. The clearinghouse includes a selection of template documents for disaster recovery contributed by various federal chief information officers as best standard practices for large enterprises.

References

1. Nemzow, M. "Business Continuity Planning." International Journal of Network Management 7 (July 1997): 127-136.
2. Ellison, R.J., D.A. Fisher, R.C. Linger, H.F. Lipson, T. Longstaff, and N.R. Mead. "Survivable Network Systems: An Emerging Discipline." Pittsburgh, PA: Software Engineering Institute, 1999: 1-3 <www.sei.cmu.org>.
3. Clarke, A.C. 2001: A Space Odyssey. Reissue ed. New York: Roc, Sept. 2000.
4. Hardy, K. "Contingency Planning." Business Quarterly 56.4 (1992): 26-28.
5. Allen, J.H. The CERT Guide to System and Network Security Practices. 1st ed. Upper Saddle River, NJ: Addison-Wesley, 2001: 447.
6. Swanson, M., et al. Contingency Planning Guide for Information Technology Systems: Recommendations of the National Institute of Standards and Technology. Diane Pub. Co., May 2004 <www.nist.gov>.
7. Quirchmayr, G. Survivability and Business Continuity Management. Eds. P. Montague and C. Steketee. Proc. of the Second Australasian Information Security Workshop, Dunedin, New Zealand, Jan. 2004.
8. Bakry, S.H. "Development of Security Policies for Private Networks." International Journal of Network Management 13 (2003): 203-120.
9. Panko, R.R. Corporate Computer and Network Security. 1st ed. Upper Saddle River, NJ: Pearson Education, Inc., 2004.
10. Whitson, G. "Computer Security: Theory, Process, and Management." Consortium for Computing Sciences in Colleges 2003.

About the Author

Ronda R. Henning is a senior scientist in the Government Communications Systems Division at Harris Corporation, an international communications company. She is the network security manager for the Federal Aviation Administration's Telecommunication Infrastructure program, a $1.7 billion network modernization of the National Air Space. Previously, she led the Harris Information Assurance Center of Excellence in defining security architectures for the National Crime Information Center and the Eastern Test Range Modernization Programs. Henning also served as principal investigator on the Network Vulnerability Visualization Architecture program, and the Integrated Design Environment for Assurance program. Prior to this, she worked in information security research and development at the National Security Agency. She is a Certified Information Systems Security Professional and a Certified Information Security Manager. Henning has a Masters of Business Administration from the Florida Institute of Technology, a Master of Science in computer science from Johns Hopkins University, and a Bachelor of Arts from the University of Pittsburgh.

Harris Corporation
Government Communications Systems Division
MS F-11
1025 W NASA BLVD
Melbourne, FL 32919
Phone: (321) 309-2642
Fax: (321) 309-2590
E-mail: rhenning@harris.com

Sixteen Standards-Based Practices for Safety and Security

Dr. Linda Ibrahim
Federal Aviation Administration

This article presents 16 standards-based practices for safety and security. These practices were derived from four safety standards and four security standards and then harmonized to recognize the commonalities among the safety and security disciplines. Implementation of these practices should lead to establishing a safety and security capability, identifying and managing safety and security risks, and assuring that products and services are safe and secure throughout their life cycle.

Safety and security are critical properties of products and services today. There are many separate standards that pertain to safety and to security covering, for example, systems, software, management, and engineering. There are also many underlying process frameworks that help organizations develop and improve their essential management and engineering processes. However, there has been a lack of alignment among both specialized standards themselves and between those specialty areas and underlying process frameworks broadly used for process improvement. To fill these gaps, the Safety and Security Extensions Project was launched, co-sponsored by organizations within the Federal Aviation Administration (FAA) and the Department of Defense.

The Safety and Security Extensions Project developed the essential practices presented in this article. The safety and security practices produced from this project were required to be based on widely recognized safety and security standards[1] and harmonized across the safety and security disciplines. After separate safety practices and security practices were derived capturing their respective source standards, the practices were harmonized to address commonalities, coordinate activities, align terminology, and encourage the merger of the safety and security disciplines. Additionally, the project provided a mechanism for implementing these specialty-engineering practices in the context of existing process improvement frameworks to align safety and security improvements with more general process improvement endeavors.

The project team comprised more than 30 experts from government and industry in the United States and the international community. The resultant practices enjoyed multiple broad national and international reviews over the two-year project duration. The final report [1] fully describes the project and the safety and security practices. That report also provides guidance regarding using these safety and security practices with integrated capability maturity models as underlying process improvement frameworks.

The purpose of this article is to spread awareness of these essential, integrated, harmonized, standards-based practices and to encourage their use in organizations concerned about safety and security. Note that these practices are not software-specific, but address a broader product and service scope. These 16 practices will form a basis for the emerging international standard, International Organization for Standardization/International Electrotechnical Commission 15026 Systems and Software Assurance.

The 16 standards-based practices for safety and security are organized by goal and summarized in Table 1, and then briefly described in the next section. Lastly, further information is provided regarding using these practices.

The 16 Standards-Based Practices for Safety and Security

Establishing a Safety and Security Infrastructure

The first five practices help establish safety and security capability and infrastructure.

Practice 1: Ensure safety and security awareness, guidance, and competency.
Those who engage in different safety and security activities need appropriate knowledge and skills. This practice includes identifying competency and awareness needs, ensuring those needs are met, and retaining records of qualifications and training. It includes qualifications required to access, use, and maintain a safe and secure work environment. The practice applies to managers, acquirers, developers, maintainers, operators, and general staff.

Practice 2: Establish and maintain a qualified work environment that meets safety and security needs.
The work environment should address safety and security needs. This includes ensuring that facilities, tools, and equipment are calibrated or otherwise qualified in accordance with appropriate standards. New technology

Table 1: *The 16 Standards-Based Practices for Safety and Security (titles only)*

Establishing a Safety and Security Infrastructure	
1.	Ensure Safety and Security Competency
2.	Establish Qualified Work Environment
3.	Ensure Integrity of Safety and Security Information
4.	Monitor Operations and Report Incidents
5.	Ensure Business Continuity
Managing Safety and Security Risks	
6.	Identify Safety and Security Risks
7.	Analyze and Prioritize Risks
8.	Determine, Implement, and Monitor Risk Mitigation Plan
Satisfying Safety and Security Requirements	
9.	Determine Regulatory Requirements, Laws, and Standards
10.	Develop and Deploy Safe and Secure Products and Services
11.	Objectively Evaluate Products
12.	Establish Safety and Security Assurance Arguments
Managing Activities and Products	
13.	Establish Independent Safety and Security Reporting
14.	Establish a Safety and Security Plan
15.	Select and Manage Suppliers, Products, and Services
16.	Monitor and Control Activities and Products

should be provided when necessary to improve the work environment.

Practice 3: Identify required safety and security information and maintain storage, protection, and access and distribution control for it.
This practice ensures that a capability exists for retaining and protecting required safety and security information. Access is controlled, and information is distributed or made available to authorized stakeholders when needed.

Practice 4: Monitor operations and environmental changes, report and analyze safety and security incidents and anomalies, and initiate corrective actions.
The operational environment is monitored, including changes in threats, hazards, vulnerabilities, impacts, and risks. Safety and security incidents and anomalies are detected, collected, reported, analyzed, and retained to assist future analyses. Analysis may lead to initiating corrective or preventive actions, risk mitigation actions, further investigation, or other actions.

Practice 5: Establish and maintain plans to ensure continuity of business processes and protection of assets.
This practice includes identifying and assessing risks to business continuity, and establishing plans to protect the business and to counteract potential business disruptions from adverse conditions, failures, and threats. Business continuity plans are tested to ensure they are up to date and effective.

Managing Safety and Security Risks

These three practices pertain to identifying and managing safety and security risks.

Practice 6: Identify risks and sources of risks attributable to vulnerabilities, security threats, and safety hazards.
This practice includes identification of security threats or safety hazards, and vulnerabilities, faults, and failures that could be exercised or exploited. Risk sources may be natural or man-made, both accidental and deliberate.

Practice 7: For each risk associated with safety or security, determine the causal factors, estimate the consequence and likelihood of an occurrence, and determine relative priority.
Causal factors for threats or hazards may include hardware, software, human and environmental factors. The severity and likelihood of an occurrence are assessed and combined to obtain an estimate of risk for each threat or hazard. The practice also addresses prioritization of these risks.

Practice 8: Determine, implement, and monitor the risk mitigation plan to achieve an acceptable level of risk.
This practice pertains to determining controls or countermeasures to reduce risks to an acceptable level. A risk mitigation plan documents the approach and activities to ensure safety and security criteria are met, and these essential risk mitigations become included as product or service requirements. Risk mitigation plans are implemented and monitored and corrective actions taken as required to control safety and security risks.

Satisfying Safety and Security Requirements

The next four practices focus on determining safety and security requirements, and ensuring they are met throughout the life cycle.

Practice 9: Determine applicable regulatory requirements, laws, standards, and policies and define levels of safety and security.
This practice addresses the determination of applicable safety and security laws, mandates, regulatory requirements, external policies, and standards. It includes establishing internal policy regarding safety and security, defining criteria for determining safety and security levels[2], and denoting methods, techniques, rules, and tools required for each level.

Practice 10: Develop and deploy products and services that meet safety and security needs and requirements, and operate and dispose of them safely and securely.
Starting from the determination of safety and security requirements and levels, this practice guides all life-cycle activities to ensure products and services are designed, developed, transitioned, deployed, operated, maintained, and disposed of to address established safety and security needs and requirements.

Practice 11: Objectively evaluate products and services to ensure safety and security requirements are achieved and products and services fulfill their intended use.
This practice addresses verification, assessment, and audit throughout the life cycle to determine whether products and services meet safety and security requirements. It also addresses validation to determine fulfillment of intended use. Evaluation activities are performed at an appropriate level of rigor as determined by required safety and security levels, and evaluation evidence is collected to support safety and security claims.

Practice 12: Establish and maintain safety and security assurance arguments and supporting evidence throughout the life cycle.
To demonstrate that safety and security assurance needs have been satisfied, it is essential to retain documentation that provides an argument for the safety and security of a product or service. This argument, with supporting evidence, is built, collected, and retained throughout the life cycle.

Managing Activities and Products

The last four practices address safety and security management so that safety and security activities and products are planned, tracked, measured, monitored, and improved.

Practice 13: Establish and maintain independent reporting of safety and security status and issues.
Reporting safety and security status and issues should reflect the degree of independence appropriate to the needed level of safety and security associated with the product or service. Staff members and external organizations need to be informed about both reporting channels and notification mechanisms.

Practice 14: Establish and maintain a plan to achieve safety and security requirements and objectives.
This practice involves establishing the plan, establishing commitment to the plan, and coordinating and communicating plans, status, and direction to participants and stakeholders.

Practice 15: Select and manage suppliers, products, and services using safety and security criteria.
This practice addresses the capability of suppliers to meet safety and security needs. Appropriate criteria are used in supplier selection, and suppliers are required to deliver safety and security assurance with supplied products and services, including off-the-shelf products.

Practice 16: Measure, monitor, and review safety and security activities against plans, control products, take corrective action, and improve processes throughout the life cycle.
This last practice broadly encompasses several process and product control and monitoring activities. It includes measuring, monitoring, and reviewing activities against plans; assuring that changes to requirements, products, plans, and procedures do

not adversely affect safety and security; taking corrective action to address any safety and security problems or issues; and improving safety and security processes throughout the life cycle.

Applying the Practices

Organizational Context

Whatever your organizational perspective, these practices are useful if you are concerned about safety and security. The practices address safety and security in several contexts, including strategically, to support enterprise-wide safety and security work; at a program-level, for any program or organization that deals with safety and security of products and services across the life cycle; in the work environment, for ensuring people have tools and facilities needed for safe and secure development, operation, maintenance, and support of products and services; and by acquisition programs, for evaluating the capability of suppliers to deliver safe and secure products and services.

Although the practices are harmonized, they can be implemented in the context chosen by the organization, which may be security only, or safety only, or both safety and security.

Process Improvement Context

The safety and security standards-based practices described above were developed to be used with respect to two integrated capability maturity models, the FAA integrated Capability Maturity Model (iCMM) [2] and Capability Maturity Model Integration (CMMI®) [3] and their respective appraisal methods. The architectural mechanism for realizing this alignment is called an *application area*. This construct was devised to ensure visibility, improvability, and appraisability of the safety and security practices, to facilitate their selective use, and to utilize the depth of existing framework practices for implementing safety and security without disrupting the underlying process frameworks already in use.

An application area groups together related standards-based *application practices* considered essential for achieving requisite outcomes particular to the application/discipline. In the case of the safety and security application area, the 16 practices presented in this article are denoted *application practices*. Application practices are carried out with associated guidance from the source standards for the application (i.e., the harmonized guidance from the eight source standards for safety and security that is provided for each practice in [1]).

However, to align the specialized safety and security practices with the essential and more general practices in the iCMM and CMMI, each application practice is further supported by *implementing practices*. These are particular practices in the CMMI and iCMM reference models that are used to implement application practices, when interpreted by the safety and security context information associated with each application practice.

Application Practice 3 -
Ensure Integrity of Safety and Security Information

Identify required safety and security information and maintain storage, protection, and access and distribution control for it.

Description: Identify required safety and security document and information. Manage and control required information, including documentation, data, and asssurance evidence to ensure its integrity. Ensure that artifacts related to safety and security assurance monitoring and evaluation are suitably protected and distributed to authorized stakeholders.

Implementing Practices: This application practice is implemented by performing the following practices in such a way as to identify required safety and security information and maintain storage, protection, and access and distribution control for it.

Process Area 17 Information Management *(from iCMM)*

Best Practice 17.01	Establish and maintain a strategy and requirements for information management.
Best Practice 17.02	Establish an infrastructure for information management, including repository, tools, equipment, and procedures.
Best Practice 17.03	Collect, receive, and store information according to established strategy and procedures.
Best Practice 17.04	Disseminate or provide timely access to information to those who need it.
Best Practice 17.05	Protect information from loss, damage, or unwarranted access.
Best Practice 17.06	Establish requirements and standards for content and format of selected information items.

Figure 1: *Presentation of an Application Practice and Its Implementing Practices*

For example, to implement application practice No. 1 *Ensure Safety and Security Competency*, implementing practices are from the training process areas in iCMM (PA 22 Training) and CMMI (Organizational Training); those practices would be carried out with particular focus on ensuring safety and security competency.

Thus an application area is structured to include the following: purpose, required goals/outcomes (application goals), expected practices (application practices), and for each application practice: description, typical work products, notes, and a list of *particular implementing practices* from the reference models. The generic practices of CMMI or iCMM apply to an application area so capability levels can be determined by appraisal methods associated with these reference models.

The excerpt in Figure 1 illustrates, for Practice 3 described in the previous section, how application practices and implementing practices are presented.

The safety and security application area, and the application area construct are fully described in [1]. The application of these safety and security practices in the context of other standards and frameworks is additionally described in [4].

Summary

This article has presented 16 essential standards-based practices for safety and security. It further illustrated how an organization can align implementation and improvement of these 16 practices using a more general process improvement model. This should lead to more efficiency and effectiveness in process improvement efforts for those organizations concerned about safety and security.◆

References

1. Ibrahim, L., et al. Safety and Security Extensions for Integrated Capability Maturity Models. Washington: Federal Aviation Administration, Sept. 2004 <www.faa.gov/ipg>.
2. Ibrahim, L., et al. The Federal Aviation Administration Integrated Capability Maturity Model (FAA-iCMM). Vers. 2.0. Washington: Federal Aviation Administration, Sept. 2001 <www.faa.gov/ipg>.
3. CMMI Product Team. Capability Maturity Model® Integration (CMMI ®), Vers. 1.1 - CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Develop-

® CMMI is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Coming Events

November 6-9
6th Annual Amplifying Your Effectiveness Conference
Phoenix, AZ
www.ayeconference.com

November 6-9
5th Working IEEE/IFIP Conference on Software Architecture (WICSA)
Pittsburgh, PA
http://sunset.usc.edu/~softarch/wicsa5

November 7-9
Airborne Early Warning and Battle Management Conference
Washington D.C.
www.marcusevansbb.com/AEW

November 13-17
International Association for Computing Machinery SIGAda 2005 Conference
Atlanta, GA
www.sigada.org/conf/sigada2005

November 14-18
STARWEST 2005
Anaheim, CA
www.sqe.com/starwest

November 14-18
5th Annual Capability Maturity Model® Integration Technology Conference and User Group
Denver, CO
www.sei.cmu.edu/cmmi/events/cmmi-techconf.html

November 14-18
3rd International Conference on Software Process Improvement (ICSPI) 2005
Orlando, FL
www.icspi.com

May 1-4, 2006
2006 Systems and Software Technology Conference

Salt Lake City, UT
www.stc-online.org

ment, and Supplier Sourcing (CMMI-SE/SW/IPPD/SS, Vers. 1.1) Continuous Representation. Pittsburgh, PA: Software Engineering Institute, Mar. 2002 <www.sei.cmu.edu>.
4. Ibrahim, L., C. Wells, and R. Bate. Extending Systems Engineering Frameworks for Special Application Areas: Case Study Safety and Security. Proc. of the 15th Annual International Symposium of the International Council on Systems Engineering, Rochester, N.Y., July 2005 <www.faa.gov/ipg>.

Notes

1. The safety and security practices are derived from the following eight standards:

   For Safety:
   - MIL-STD-882C*. Military Standard System Safety Program Requirements. U.S. Department of Defense, Jan. 1993 <https://safety.army.mil/pages/systemsafety>.
   - MIL-STD-882D*. Standard Practice for System Safety. U.S. Department of Defense, Feb. 2000 <https://safety.army.mil/pages/system safety>.
   - IEC 61508. Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. International Electrotechnical Commission, 1997 <www.iec.ch/61508>.
   - DEF STAN 00-56. Defence Standard 00-56, Safety Management Requirements for Defence Systems. Ministry of Defence, United Kingdom, Dec. 1996. <www.dstan.mod.uk> or <http://cms.brookes.ac.uk/modules/other/58_DEF_STAN_00-56.pdf>.

   * Although MIL-STD-882D supercedes MIL-STD-882C, the knowledge in MIL-STD-882C was also integrated into the 16 practices, as proposed/endorsed by the safety community. We found no inconsistencies between the standards and included them both.

   For Security:
   - ISO/IEC 17799:2000(E). Information Technology – Code of Practice for Information Security Management. 1st ed. International Organization for Standardization, 1 Dec. 2000 <www.iso.ch>.
   - ISO/IEC 15408. Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements, Vers. 2.1. Common Criteria Project Sponsoring Organizations, 1999 <www.commoncriteriaportal.org>.
   - ISO/IEC 21827:2002. Systems Security Engineering Capability Maturity Model. International Organization for Standardization. (Systems Security Engineering Capability Maturity Model, Model Description Document Vers. 3.0, June 2003, Systems Security Engineering Capability Maturity Model (SSE-CMM) Project.) <www.sse-cmm.org> or <www.iso.ch>.
   - NIST 800-30. Risk Management Guide for Information Technology Systems. National Institute of Standards and Technology, Special Publication 800-30, 2001 <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>.

   The Safety and Security Extensions final report [1] includes mappings of the safety and security practices to source practices/clauses, and demonstrates coverage of these eight source documents, at an appropriate level of detail.
2. Each safety and security level denotes an expression of trust or an acceptable range of values of risk containment associated with a product or service.

About the Author

Linda Ibrahim, Ph.D., is chief engineer for Process Improvement at the Federal Aviation Administration (FAA). She led development and is lead author and architect of FAA-integrated Capability Maturity Model Vers. 1.0 and Vers. 2.0, and its appraisal method, and she co-managed the Safety and Security Extensions project. Ibrahim has worked in software engineering for more than 30 years in the United States, Europe, and Middle East, and is a member of the Capability Maturity Model® Integration Steering Group. Ibrahim has a Bachelor of Arts in Mathematics, a Master of Science in information science, and a doctorate in electrical engineering.

Federal Aviation Administration
800 Independence AVE SW
Washington, DC 20591
Phone: (202) 267-7443
Fax: (202) 267-5069
E-mail: linda.ibrahim@faa.gov

The Information Technology Security Arms Race

Dr. Steven Hofmeyr
Sana Security

Increasingly, new attack technologies and tools are overwhelming existing information technology defenses. This ongoing arms race requires new technologies for the defender. In this article, I describe how an intrusion prevention system (IPS) fills this need, and I dissect the sometimes confusing world of IPS, describing the various technologies available, and where and how to deploy them.

Security for information technology (IT) is subject to an ongoing arms race between the attackers and the defenders. As new IT systems are developed and deployed, attackers find new weaknesses in these systems and new ways of exploiting the weaknesses. Defenders must keep innovating to keep up with the attackers; without ongoing innovation, IT systems will become crippled by attacks.

New Attack Technologies

Recently, attackers have devised a set of new tools to make it easier to attack systems and evade defenses. A good example is a group of tools known as *binary differs* that determine differences in binary code [1]. Binary differs are typically used to determine the difference between a patched version of an application and an unpatched version. This enables the attacker to determine what vulnerability was patched, and how.

Armed with this information, an attacker can rapidly develop an exploit for that vulnerability, often within a matter of hours. Consequently, as soon as a new patch is announced, an attacker can have an exploit for that patched vulnerability within a matter of hours. This is a problem because defenders can rarely, if ever, patch that fast, even with automated patch systems. Patches have to be tested before being deployed and that can take days, even weeks. With the advent of binary differs, the defender will lose the patching race every time. Patching is no longer a viable defense strategy.

Another kind of tool is the automated attack framework. With these frameworks, an attacker with little or no technical expertise can easily launch a variety of attacks. The best example is the free open source tool Metasploit [2], which allows an attacker to scan a system for vulnerabilities, and then gives the attacker the choice of various attack modules to click on to launch an attack. In addition to choosing the type of attack, the attacker also gets to choose the type of payload, which can be any one of a variety of malicious software (malware for short) such as a Trojan horse or a root kit. Such automated attack frameworks make it very easy for anyone to launch attacks on systems and to encourage the rapid dissemination of attack information.

Not only is it much easier to launch attacks, but the payloads of those attacks are becoming increasingly sophisticated. We are seeing a proliferation of malware designed to do nasty things to the victim such as stealing information, spreading rapidly, and clogging networks, and serving as bot[1] networks that can be used to launch denial-of-service attacks or function as spam relays.

“Before harm is done, an IPS [intrusion prevention system] will proactively detect attacks and prevent them, providing a powerful new layer of defense.”

Often this malware will be tailored to a particular attack, for example, a Trojan horse may be specifically designed to steal information from one organization, to be used only in that circumstance. This means that Trojan horse will be something new and not detected by signatures in traditional antivirus or antispyware tools. If a signature is developed (assuming the Trojan horse is ever discovered), it will be useless because the Trojan horse will never be used again.

It is a trivial matter to develop customized malware: in the simplest case, the attacker can use Morphine [3], a hosted service that will obfuscate any piece of malware for a small fee (about $36), ensuring that the malware is not detectable by any of the standard signature-based antivirus systems.

New Technologies for Shielding Vulnerabilities

With the new technologies attackers are developing, we can no longer rely on reactive technologies such as patching and static signature scanning. These technologies are too slow to prevent widespread damage, and too dependent on human expertise to be able to scale to the complexity and growing size of IT systems today. We need a new approach.

We can move forward in the arms race by using the proactive technology in an intrusion prevention system (IPS). Before harm is done, an IPS will proactively detect attacks and prevent them, providing a powerful new layer of defense. An IPS can be used to shield vulnerable systems, giving time for administrators to fully test and deploy patches at their own convenience, and buying time for human operators to better understand the threat. Further, an IPS reduces the need for human expertise and saves on expensive and hard-to-scale human resources. Properly deployed, IPS can protect against a wide variety of threats, including surreptitious malware and fast-spreading destructive worms and viruses.

The world of IPS technology can be very confusing. There are a host of different technologies available, and it can be very difficult for administrators to determine which are the most suitable for their needs. To add to the confusion, IPS can mean different things to different people. In this article, I discuss the relatively new technologies (within the last few years) that are commonly acknowledged to comprise an IPS. I do not include technologies such as firewalls and signature-based antivirus systems that have been around for many years and are failing to secure IT systems. The intent of this article is to clarify the IPS landscape; to this end, there are three aspects that need to be considered: (1) where to deploy IPS, (2) what kind of IPS technology to deploy, and (3) how to deploy the chosen IPS system. I will address each of these in turn.

Where to Deploy IPS

There are basically two places to deploy IPS: on the network or on the host computer (see Figure 1). The network IPS has several advantages: It is usually a single device or appliance that is both easy to manage and easy to deploy. All the security administrator need do is drop a box onto the network segment; usually no permission is required from the application owners because there will be no conflicts with software installed on the hosts. Furthermore, a network IPS provides broad coverage if placed at an appropriate choke-point: A single appliance can protect a whole segment with multiple hosts.

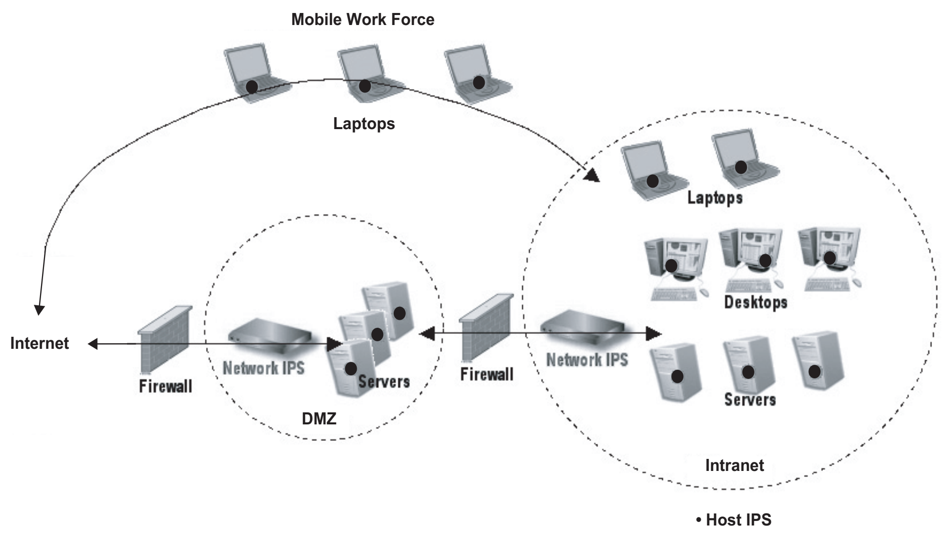
However, there are limitations to the network IPS. The broad coverage can also be detrimental because failure of a single appliance will cut off traffic to a whole subnet, a consequence of the fact that the IPS has to be inline to be able to drop malicious traffic. There is also a performance tradeoff because the more hosts a single appliance protects, the more traffic it will have to process; a network IPS that does sophisticated traffic analyses can rapidly become a bottleneck, unable to cope with high traffic volumes. When deploying network IPS, an administrator should be well aware of this tradeoff.

By contrast, IPS on the host does not suffer from the same problems. Each host will have its own IPS software so the security processing is distributed across all machines and performance is no longer an issue. Further, a host IPS tends to be more robust, because failure of one system will only have a small affect on the overall performance of hosts in the network.

There is another powerful driver toward using host IPS: de-perimeterization. Network IPS requires a clear notion of a network perimeter, and unfortunately the perimeter is collapsing with the advent of distributed applications – such as Web services – and more business being done over the Internet necessitating closer links with partners, suppliers, etc. This trend is so powerful that a high-level industry organization, the Jericho Forum, has been created to promote de-perimeterization [4]. The loss of the perimeter is exacerbated by the increasingly mobile work force. Users that work from outside the corporate or government network can easily pick up malware infections and bring those into the secure environment, infecting all vulnerable hosts behind the firewall. This is another compelling reason for the rapid adoption of host IPS.

In typical deployments, however, an organization will use both network and host-level IPS. This layered approach generally gives the most comprehensive security, although organizations deploying multiple layers should be aware that the more layers in place, the more chance there is of false positives, and the more difficult it is to manage the system. Any security architecture using IPS will also include network-level defenses to protect against network-level threats such as denial-of-service attacks and eavesdropping, but these are not generally considered part of IPS, and are not discussed in detail in this article.

Figure 1: *Typical IPS Deployment*



What to Deploy: IPS Detection Technologies

Although IPS is designed to prevent attacks, and not just detect them, it is still reliant on its underlying detection technology: Only that which is detected can be prevented. Attacks are detected and then prevented by an IPS. For example, a network IPS that drops packets[2] from an attack has first detected the packets and then prevented the attack by dropping the packets. Similarly, a host IPS that prevents applications from making system calls has detected the system calls being attempted, and prevented them from being executed, hence preventing the attack. The discussion of IPS technology is greatly clarified by separating out the detection from the prevention aspects. In this section, only detection technologies are discussed; in the section, "What to Deploy: IPS Prevention Technologies," prevention technologies are discussed.

There are a variety of detection technologies available, each with its advantages and disadvantages. Many of these have long been in development and used in intrusion detection systems, and are little changed although they are used in an IPS. Often the best solution is a layered approach in which multiple technologies are used to complement each other's strengths and cover each other's weaknesses.

Signature-Based Detection

Signature-based detection relies on human experts knowing and understanding what particular exploits look like so the experts can encode signatures for those exploits. Typically, signature-based technology is used to scan static data such as network packets. Signatures are *exploit-focused*, meaning they will not protect a vulnerability, but only particular exploits of that vulnerability. For example, there are many ways to write a buffer overflow, but typically a signature will only look for one way of doing so; if the attacker changes the code in the exploit, the signature will not detect it.

Generally, there are many different ways in which a vulnerability can be exploited, and a signature will only protect against one of those. Because of this, signatures only protect against yesterday's attacks – those that we already know about. Signatures will not protect against zero-day (unknown) threats, or against modified or mutated malware. In addition, signatures require extensive human expertise and constant updating, an approach that does not scale well. Fundamentally, signatures are one of the poorest choices for detection in IPS because of the human overhead, and the fact that this approach fails continually. For these reasons, signatures are rarely used in IPS.

Expert-Based Detection

Expert-based detection relies on human experts defining a set of rules that dictate what behaviors are normal, and hence, allow for applications and operating systems. The expert-based approach relies on humans understanding the applications and systems to be protected, but requires little or no knowledge of attacks. This approach can be very effective at protecting against zero-day threats because these generally cause deviations in application or system behavior. However, this approach requires that the people defining the rules know a great deal about the applications and system to be protected, which can be problematic for complex and custom applications.

When an application is complex, expert-based detection ends up being detuned to reduce false positives: The rules become increasingly generalized to the point where they offer little or no protection at all. For example, an IPS might have a rule preventing applications from loading drivers into the kernel. However, some applications may legitimately need to load drivers, which would result in a false positive. A common response is to turn off the rule altogether because of the complexity of determining which applications should be allowed to load drivers. Relaxing the rule will allow any application to load drivers with consequent security risks.

Generally, the expert-based approach works best when applied to well known, relatively simple applications with well-known behavior that does not deviate significantly from one system to the next, or from one use to the next.

Specification-Based Detection

Specification-based detection gets away from dependence on human expertise by deriving a set of rules governing normal (allowed) behavior by either statically scanning protocols, or application binary or source code. This has the advantage of avoiding the overhead of human involvement and human bias and error, which can plague approaches such as expert-based detection. Although this approach can be powerful, it is limited in that it is often too generalized.

When protecting applications, the protection will usually only prevent injected code, and not detect (and hence be able to prevent) the large number of attacks that exploit other types of vulnerabilities such as misconfigurations. In the case of protocol-based network protection, it is limited in that it cannot detect flaws in the protocol itself, and often networking components and applications will not implement protocols strictly according to the specifications, resulting in many false positives.

Autodidactic Detection

Autodidactic detection attempts to automatically derive a normal model like the specification-based approach, but uses learning or auto-configuration to refine the normal profile so that the approach does not suffer from excessive generalization. The normal model is derived from monitoring systems during normal usage and learning the normal model of network traffic, or host behaviors. This learning can be done in a production environment or a quality assurance (QA) laboratory.

"Where an organization is uncertain, they should follow a process whereby they first test the performance of the IPS. On the host, this means measuring the impact on running applications and memory/disk usage, and in the network, this generally means measuring network throughput and latency."

Autodidactic systems tend to be scalable and offer very good protection for all kinds of applications and networks because of their ability to automatically learn all the nuances of complex behavior. However, there are several limitations to this approach. First, only stable behaviors or network patterns can be learned, so in highly variable environments or with highly variable usage patterns the system can prove to be inaccurate. Second, learning takes time, during which the system can be vulnerable to attack. This can be offset by doing the learning in another environment such as a QA lab or a test lab, but how well that works depends on how closely the lab environment represents the real environment.

Innate Defense Detection

Innate defense detection is most similar to signature-based in that it protects against specific attacks, but it differs from signature-based in that it is *vulnerability-focused*: It detects a whole class of attacks rather than instances of a particular class. A good example is technology for detecting buffer overflows: There are many different attacks that exploit buffer overflows, necessitating many different signatures, but an innate defense for buffer overflows will stop every kind of buffer overflow without requiring any knowledge of specific attacks.

Generally, an innate defense will detect one class of threats, and do so effectively, with high accuracy and little or no overhead in terms of tuning or configuration. This approach is very important for protecting systems by default on a large scale since this kind of technology can easily be distributed with the operating system. It can be used stand-alone to incrementally improve protection, but because the protection provided by innate defenses is not comprehensive, it is best used as part of a layered solution.

The major shortfall of this approach is that not all threats are amenable to prevention through innate defenses; there may not be any way of developing a generic method of detecting all attack instances within a given threat class.

What to Deploy: IPS Prevention Technologies

The hardest aspect of prevention is that it should be immediate (proactive) whenever possible. This can be problematic if the system has to initially gather information to determine if an attack is actually happening; too much delay in the response will result in damage. There is often a trade-off: The more information is used to determine if an attack is happening, the more accurate the IPS, but the more potential there is for the attack to cause harm. Another important aspect of prevention is that it can cause damage when reacting to false positives because it can block legitimate behavior or data. One way to minimize the potential harm is to have responses that are as fine-grained as possible, for example, to block a particular system call, rather than kill a process.

However, fine-grained responses can only go so far in ameliorating the damage done by responses to false positives. The problem is that currently all prevention tends to be all or nothing: Either an action is allowed, or it is blocked. For example, host IPS tends to block actions such as file accesses, network accesses, process starts, etc., and network IPS tends to drop packets, reset connections, and block particular Internet protocol addresses. To move forward, we need *benign*, recoverable responses. The goal is responses that can stop attacks, but allow the system to recover from false positives.

For example, databases have mechanisms for rolling back transactions; we can expect to see similar concepts in the future in IPS. Another method of benign response is using delays to slow down the rate at which attacks propagate. Research has shown that this can be an effective method of blocking attacks [5], and can also be very useful in slowing down virus and worm propagation [6]. Prevention technologies in IPS today are extremely useful and powerful, but we expect to see great improvements in the future.

How to Deploy IPS

Typically, the first step in deploying IPS will be testing and evaluation. The extent to which an organization does testing depends on many factors, including the resources available for testing and how comfortable an organization is with the reported and claimed functioning of IPS. Where an organization is uncertain, they should follow a process whereby they first test the performance of the IPS. On the host, this means measuring the impact on running applications and memory/disk usage, and in the network, this generally means measuring network throughput and latency. Stability can be an even more important measure: Does the IPS crash and block network traffic or bring down the host? A good IPS should never impact performance and stability unreasonably, although bear in mind that network IPSs are usually designed to support various speeds, and an IPS that processes traffic faster tends to be a lot more expensive.

Another important factor to test is accuracy. Does the IPS stop attacks effectively without high false alarm rates? One way to test this is in a lab by running the IPS on vulnerable systems and actively launching attacks against them. Although this gives some idea of accuracy, it is limited in that it gives very little idea of usability, scalability, and false positive rates in the real world. For example, it will often be a simple matter to configure an expert-based IPS in a lab, protecting standard applications – the accuracy may appear to be very high. However, deploying such technology out in the production environment may lead to many false alarms with a subsequent detuning and loss of accuracy of the IPS. The importance of evaluation in the production environment cannot be overstated.

Once the IPS is tested, the next phase is usually a limited deployment phase, using the IPS to protect a few systems such as those under high threat in the Demilitarized Zone[3], or those that are of little importance (hence false positives are acceptable). When the IPS has proven itself in the limited deployment phase, it can then be deployed across the whole organization. This process will obviously differ for network and host IPS.

Once deployed, the organization enters the maintenance phase; most IPS will have to be tuned whenever there are changes in the organization such as additions of new networks or machines, or reconfigurations of applications. An organization should plan for these changes knowing that they will have an effect on the IPS deployment. As stated before, some IPS technologies are much more adaptable than others (for example, the autodidactic approach) and are much easier to deal with during the maintenance phase.

"Once deployed, the organization enters the maintenance phase; most IPS will have to be tuned whenever there are changes in the organization such as additions of new networks or machines, or reconfigurations of applications."

The Benefits of IPS

IPS is starting to be widely deployed in many different market sectors across many different organizations. There are three markets that are seeing immediate benefit from IPS: the financial sector; the government, including the military; and the health-care industry. All of these sectors are under increasing attack and feeling the pressure of new government regulations such as Sarbanes-Oxley [7], the California Senate Information Disclosure Bill 1386 [8], and the Health Insurance Portability and Accountability Act of 1996 (HIPAA) [9].

For example, military organizations are using the Internet and commercial off-the-shelf software to realize efficiency gains, but are consequently at risk from attacks that target platforms such as Microsoft Windows. It is vital to the security of the country that these organizations maintain high standards of protection; to this end, IPS is an essential part of the defenses. We are also seeing the potential for IPS in the mobile battlefield where unprotected mobile computers such as laptops would be prime targets for attack, especially if they are not used and not updated when in storage and then brought out rapidly for battlefield deployment.

Financial organizations are also among the early adopters of IPS. They find IPS particularly useful for securing unpatched applications. For example, many financial organizations mandate at least three weeks' testing of any new patch because faulty patches can bring down mission-critical servers. However, three weeks of exposure for vulnerable servers connected to the Internet will almost certainly result in compromise, so these organizations turn to IPS to enable them to adequately test patches while still ensuring their servers are protected.

In the health-care industry, regulations such as HIPAA require health-care providers to ensure the confidentiality of patient data. Deploying intrusion detection systems and other reactive technologies that only determine when an attack has happened after the fact is not sufficient because valuable patient data will already be stolen. Hence, the health-care industry is realizing critical benefits through the ability to stop information leaks before any harm is done.

Summary

IT security is an ongoing arms race. Recently, attackers have been gaining the upper hand with a new set of attack tools and techniques. IPS regains the initiative for defenders, providing a shield for unpatched vulnerabilities. This buys time to test and deploy patches, reduces human resource cost, and reduces security breaches and the associated costs.

But IPS can be complex. An organization should know where to deploy IPS, whether on the host or network (ideally, both), and the tradeoffs inherent in such a decision. Further, an organization should understand what sorts of technology are available and what are most suitable for its environment. In general, the best approach is a layered one that uses multiple technologies. Finally, an organization should plan for a phase of testing and evaluation, and should know how to go about rolling out the IPS.

The technologies described in this article all exist in commercial products, of which there are many. When considering deploying IPS, an organization should search for vendors in the IPS arena and solicit information from a set of vendors to ascertain exactly what they do. This article is intended to be a useful guideline in cutting through the marketing language and enabling users to understand exactly what a vendor's products are likely to achieve.

Implementing IPS will take effort and money, without doubt, but IPS is essential in today's threat environment. Without improved security measures, our IT systems will soon become worse than useless, and the costs of failed security will far outweigh the costs of IPS.◆

References

1. Flake, Halvar. "More Fun With Graphs." Blackhat Federal 2003, Tyson's Corner, VA, 1-2 Oct. 2003 <http://cansecwest.com/csw04/csw04-Halvar.ppt>.
2. Metasploit. 26 July 2005 <www.metaspoloit.com>.
3. Anti-Detection Service. 11 July 2005 <http://hxdef.czweb.org/antidetection.php>.
4. The Open Group. The Jericho Forum. 11 July 2005 <www.opengroup.org/jericho>.
5. Somayaji, A. "Operating System Security and Stability Through Process Homeostasis." Doctoral Diss. University of New Mexico, 2002.
6. Williamson, M., J. Twycross, J. Griffin, and A. Norman. "Virus Throttling." Technical Paper HPL-2003-69. Hewlett Packard Labs, 2003.
7. Sarbanes-Oxley. "Financial and Accounting Disclosure Information." Huntington Beach, CA: Sarbanes-Oxley <www.sarbanes-oxley.com>.
8. Sen. Peace, Assembly Member Simitian. "SB 1386." California Senate, 26 Sept. 2002 <http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html>.
9. U.S. Department of Health and Human Services. "Office for Civil Rights – HIPAA." Washington: HHS, 1996 <www.hhs.gov/ocr/hipaa>.

Notes

1. A *bot* network is a network of compromised computers that are remotely controlled by an attacker. Each computer runs *bot* software that enables an attacker to access the computer and control it remotely.
2. This means not forwarding suspicious packets of network data.
3. This is the part of the network that contains Internet facing servers, and is usually separated out from the main intranet, forming a buffer zone between the intranet and the Internet.

About the Author

Steven Hofmeyr, Ph.D., is chief scientist at Sana Security, which he founded in 2000. Sana Security is a market leader in intrusion prevention with its host-based products widely deployed throughout industry and government. Hofmeyer has also carried out research at the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology (MIT) and the Santa Fe Institute for Complexity Studies. He has authored and co-authored many published papers on computer security, immunology, and adaptive computation. He has been an invited participant to several U.S. government workshops on future directions for technology such as the Joint Engineering Team Roadmap Workshop. In 2003, MIT's *Technology Review* named him as one of the top 100 young innovators under 35, and in 2004, he was named one of the 12 innovators of the year by *InfoWorld*. Hofmeyr has a doctorate in computer science from the University of New Mexico.

Sana Security
2121 El Camino Real
STE 700
San Mateo, CA 94403
Phone: (650) 292-7152
E-mail: steve@sanasecurity.com

The MILS Architecture for a Secure Global Information Grid

Dr. W. Scott Harrison, Dr. Nadine Hanebutte, Dr. Paul W. Oman, and Dr. Jim Alves-Foss
Center for Secure and Dependable Systems

Multiple Independent Levels of Security and Safety (MILS) is a joint research effort between academia, industry, and government to develop and implement a high-assurance, real-time architecture for embedded systems. The goal of the MILS architecture is to ensure that all system security policies are non-bypassable, evaluatable, always invoked, and tamper-proof. Using these formally proven security policies guarantees information flow control, data isolation, predictable process control, damage limitation, and resource availability. As applications are not considered trustworthy components, information flow control needs to be performed by entities external to the applications. This approach allows for the integration of legacy applications that do not necessarily have security integrated into them. Therefore, the MILS architecture creates an environment that adds safeguards to previously insecure applications, allowing the integration of possibly insecure applications into a secure environment. To accomplish this in the MILS architecture, guards are placed between communicating entities to act as message content filters and enforce information flow control. This article discusses issues concerning design and implementation of MILS components for message routing and guarding on a secure Global Information Grid facilitating net-centric warfare and defense.

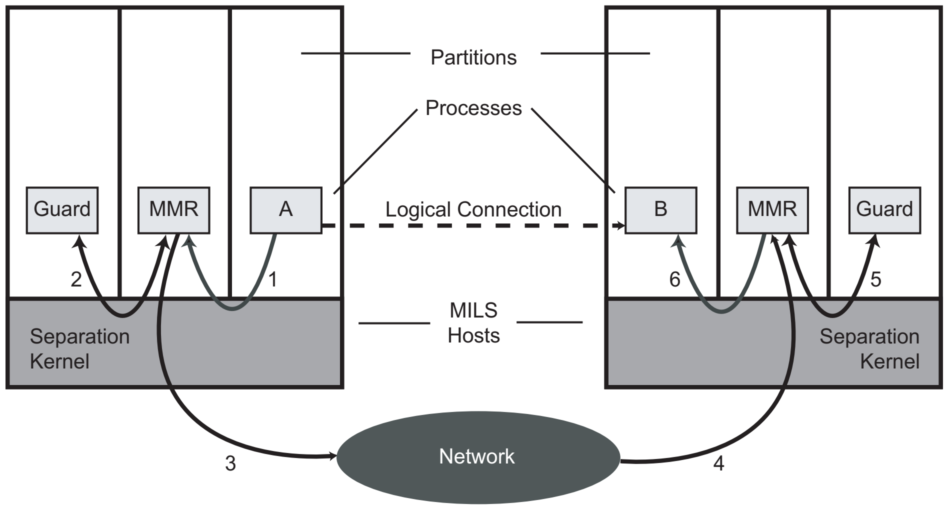
High-assurance systems are used in environments where failure can cause security breaches or even a loss of life [1]. Examples include avionics, weapon controls, intelligence gathering, and life-support systems. Before such a system can be deployed, there must exist convincing evidence that it can support critical safety as well as security properties.

The avionics community has addressed the need for safety-critical systems by developing the DO-178B and DO-255 standards, which provide a set of guidelines for the design, analysis, and evaluation of system safety [2, 3]. Though adequate for the safety evaluation of airborne systems, neither is sufficient to address the security concerns of critical security systems such as those that protect national security. Such high-assurance systems require the rigorous specification and implementation requirements outlined in the Common Criteria (CC) [4].

The CC is a jointly developed evaluation standard for software that was created by a consortium representing the United States, United Kingdom, Germany, France, Canada, and the Netherlands. The purpose of the CC is to standardize evaluation of security features in software, which allows, for example, the comparison of different security solutions. The CC achieves this by providing guidelines for the design, analysis, and evaluation of critical systems defined at seven Evaluation Assurance Levels (EALs). The higher the assurance level, the stricter the requirements mandated by the CC. At the highest levels (EAL 5-7), the CC requires the use of formal methods, mathematical models, and proofs [1].

The level of difficulty and complexity of formal verification increases in an exponential manner with the number of analyzed lines of code (LOC). Code bases of over 100,000 LOC are considered to be unverifiable [5]. The goal for a verifiable software component is under 4,000 LOC [6]. With this restriction on code destined for EAL 5 certification or higher, the design shift from monolithic code bases to smaller modular components must occur.

Figure 1: *The MILS Architecture*



A system designed specifically for EAL 5-7 certification is the Multiple Independent Levels of Security and Safety (MILS) architecture [7]. The MILS approach toward meeting the formal evaluation requirements of the CC is to separate system functionality into smaller, individually verifiable components. The MILS architecture enables the enforcement of system-wide information control policies via mechanisms built into the kernel as well as middleware components that create the authorized communications paths between applications.

One example of MILS middleware security component is an application-level message filter called a guard, or mediator. Since MILS guards can be verified independently with respect to other components, they can be built once and used within any MILS system that needs application-level message filtering.

This article describes the MILS initiative led by the Air Force Research Laboratory (AFRL) with stakeholder input from the Air Force, Army, Navy, National Security Agency, Boeing, Lockheed Martin, Objective Interface Systems, Green Hills Software, Lynux Works, Wind River, General Dynamics, Raytheon, Rockwell Collins, MITRE, and the University of Idaho. MILS technology is targeted toward the C-130 Avionics Modernization Program, F/A-22, F-35, C-130, Commanche, global positioning system, the Joint Tactical Radio System and the LandWarrior Program [8]. A prototype proof-of-concept of the system described in this article has been implemented within an embedded system at the University of Idaho.

This technology is essential for the Global Information Grid (GIG). The GIG is envisioned as a globally connected set of computer systems and software [9]. Object-oriented communications protocols (such as those we describe in this article) are vital to such a system [10]. Further, as the information on the grid will consist of many security classification levels, it will be absolutely necessary to control the information that flows through the GIG. The MILS architecture allows exactly that.

The MILS Architecture

MILS is a verifiable, secure architecture for executing different security-level processes on the same high-assurance system. The MILS architecture accomplishes this by providing two types of separation. MILS enforces a separation policy that strictly controls communication between processes of different security levels. This prevents, for instance, a top-secret process from communicating with an unclassified process. Further, MILS separates traditional kernel-level security functionalities into external modular components that are small enough for rigorous evaluation using formal methods. Verifiable secure systems can then be built from multiple, independently developed and certified components.

The foundational component of MILS is the separation kernel (SK). The SK segregates processes and their resources into isolated execution spaces called *partitions*. Processes running in different partitions can neither communicate nor infer each other's presence unless explicitly permitted by the SK. The SK enforces compliance to information flow policies via the MILS message routing (MMR) component. The primary function of the MMR is to route communication between applications in different partitions if that communication is allowed by the policies of the system [11]. If not, the MMR will not permit communication between the partitions.

In conjunction with the MMR, which simply fulfills routing functionalities on messages between partitions, guards enforce detailed, protocol-specific policies. A guard exists for each application-level protocol supported in a MILS system. If a guard determines that the content of a message does not comply with information flow policy, the guard will notify the MMR that will then disallow the communication attempt or take action based on security policy. Steps one through six in Figure 1 show the path of a message within a MILS architecture from the sending process (A) to the receiving process (B).

The advantage of using the MMR and guards is that the system does not have to trust the applications to conform to security policies. The MMR (and later, the guard) will enforce these policies. Thus, it is possible to have a secure MILS system while running untrusted applications within the partitions. This is because the SK prevents any other possible partition communication.

Because of the separation of responsibilities between message routing and message content filtering for each protocol, the individual components (the MMR and multiple guards) can be independently verified. Verification is only possible because the guards and MMR have distinct and well-defined functionalities. Neither accidental nor malicious communication attempts that violate system policy will be successful.

"The MILS architecture enables the enforcement of system-wide information control policies via mechanisms built into the kernel as well as middleware components that create the authorized communications paths between applications."

Many communication protocols were not designed to provide artifacts that allow systematic security policy violation handling such as proper error messages. In general, most protocols were not written with security as an objective, and thus, there are typically no error messages that are security-specific. As an example, consider two clients: a client classified as Secret requesting top-secret information, and a client classified as top-secret requesting the same information. Further, assume that there is an error from both clients in the request message (perhaps the query was incorrectly formed). Generally in this situation, a single error message would be returned to both clients. However, in many systems, this would not be correct behavior; we may not wish for the secret client to know that it was contacting a valid server at all, as this gives the client information about the state of the system that might invalidate security policy. Thus, there is a need, not generally implemented in most protocols, for security-specific error messages that do not reveal information about the state of a system.

Therefore, the MMR also has to determine which error messages are relayed back to the sender due to security policy violations without disrupting the overall execution of the communication initiator. A challenge similar to that of the proper relaying of error messages is that of integrating legacy software into a MILS system. This is because one of the design goals of a MILS architecture is transparency, i.e., existing (legacy) applications can be seamlessly integrated into a MILS environment. The MMR and communication channel guards facilitate this transparency.

Multi-Level Access Control

Multi-level secure (MLS) systems enforce a high-level, inter-partition security policy that dictates whether partitions with different clearances can communicate. Traditionally, the military model of a secure operating system includes a MLS concept. The idea behind this concept is that the system will be processing data items that are classified at different levels of security, and the information flow security policy that prevents the transfer of high-level classified information into low-level objects must be preserved. Therefore, we define a MLS system as one that must be certified to process and output data at multiple classification levels. Classic security models such as the Bell-LaPadula model [12] have been created to specify the secure behavior of such MLS systems. The problem with pure MLS systems is that they must be rigorously analyzed for security before they can be certified. Every portion of the MLS system must be analyzed to ensure that it properly handles labeled data and that there is no possible violation of the security policy. Even with a Trusted Computing Base architecture or reference monitor in place, there is often too much to evaluate.

The MILS architecture was developed to resolve the difficulty in certifying MLS systems by separating out the security mechanisms and concerns into manageable components. A MILS system isolates processes into partitions, which define a collection of data objects, code, and system resources. These individual partitions can then be evaluated separately. This divide-and-conquer approach will exponentially reduce the proof effort for secure systems. For instance, a traditional MLS policy may allow a secret partition to send messages to a partition that has both *secret* and *top-secret* clearance. An additional application-specific transport policy (i.e., a protocol-specific guard) is needed to do this. The protocol-specific guard policy specifies constraints on the contents of messages sent between partitions already allowed to communicate by the MLS policy.

The MMR and SK can fulfill MLS policies, while the protocol guards enforce application-specific security policies that may or may not be MLS policies; however, the SK and MMR do work in tandem with the guard policies to provide fine-grained access control of application-level messages.

A MILS Database Server

Consider a multi-level secure database application. This database would contain entries of different security levels, e.g., top secret, secret, classified, and unclassified. Remote processes from different partitions can invoke read and write methods on this central database. However, if a client process that is only classified to handle *secret* data (e.g., Secret_Read()) attempts to invoke a read method on *top-secret* data (e.g., TopSecret_Read()) from the database server, the request must either be denied or the data must be downgraded from top-secret to secret prior to invoking the requested read.

Polyinstantiation is another solution to this problem; however, the challenges are similar. Stated simply, polyinstantiation is a situation in which users at different security classifications receive (possibly) different responses to the same queries. As an example (adapted from [13]), consider that we might have information regarding a ship (S), an objective (O), and a destination (D). When an unclassified user queries this information, he or she would receive the information {S=U.S.S. Starfish, O=Surveying, D=Hawaii}.

"Thus, it is possible to have a secure MILS system while running untrusted applications within the partitions. This is because the SK [separation kernel] prevents any other possible partition communication."

However, a top-secret level user would receive the information {S=U.S.S. Starfish, O=Spying, D=Coast of Vietnam}, which presumably is the actual state of the system. Such a solution comes at the cost of having to create very large databases and, as above, requires authentication of the true originator of a request.

Adding functionality to the database partition to determine the true origin of the request sender and to verify that the sender has proper classification is a less complex solution. Such a solution, however, would cause other problems:

a. The server's code base might become too large to evaluate formally.
b. Dedicated server processes will have to be rewritten to account for every type of data transaction (e.g., top secret, secret, unclassified, etc.).
c. The server's responses to valid but unauthorized requests would need to be added to the server's code base.

The MILS solution is to allow the MMR to parse the message before it reaches the client, consult a policy, and then either pass or reject the message after an in-depth content analysis (which would potentially still be necessary, although not as thorough, for a system that incorporates polyinstantiation). Such an analysis would have to account for routing policies and protocol or content-specific policies. This requires extra complexity in the MMR, which increases with every application-layer protocol supported in the MILS system. Therefore, the MMR simply determines if the communication between partitions is allowed according to the security policy and, if so, identifies the message type and passes it on to the protocol-specific guard.

When a protocol guard receives a message from the MMR, it parses the message, consults a protocol-specific policy, and then notifies the MMR whether to allow or deny the message. If the message is allowed by the protocol policy, the MMR sends the message on to its destination. Otherwise, the MMR will perform error handling. Figure 2 shows the logical structure of the MMR and the protocol-specific guard situated between client and server applications.

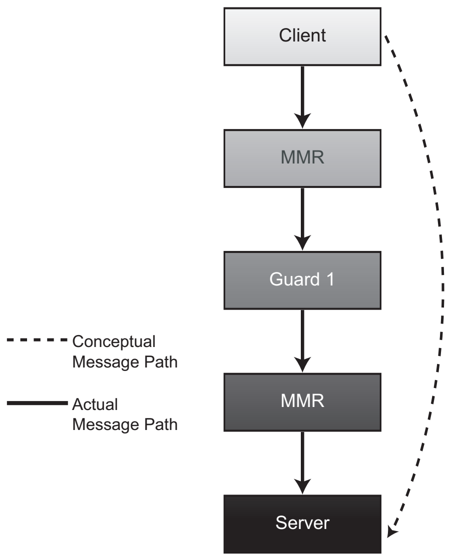
To illustrate the previous example within a system that contains a MMR and protocol guard, assume that a secret level client attempts to invoke the read method of the top-secret database server. The client encapsulates the request in a protocol-specific message and sends it. The MMR determines that the client is allowed to communicate with the server, recognizes the message as being of a particular protocol type, and routes it to the appropriate guard for analysis. The guard examines the request message and determines that the client is not allowed to invoke the read method of the top-secret database object. Finally, the guard instructs the MMR to discard the message, and possibly generate an error message to be returned to the client.

It should be noted that simply discarding the message will generally be insufficient. As the client never receives a response from the server, the client will likely continue to make the same request. Since this condition also occurs under normal circumstances such as those due to temporary network outages, for example. Thus, one function of the protocol-specific guard is to generate error packets, which look like standard protocol error messages that will be sent back to the client. If no response is expected, obviously an *error* message need not be returned.

An Example Policy

The Common Object Request Broker

Figure 2: *Processes, the MMR, and Guard Message Path*



Architecture (CORBA) is a platform-independent middleware architecture that facilitates common client/server requests. CORBA Object Request Brokers communicate via the General Inter-Orb Protocol (GIOP). We will illustrate the MILS architecture with an example using a CORBA GIOP guard.

Figure 3 illustrates an example of the MMR and a protocol-specific guard (in this case, a CORBA GIOP guard) working together to enforce MLS and transport policies. Client A is a CORBA client application running in *secret*-level partition 1, and client B is a CORBA client running in *unclassified*-level partition 2. A CORBA database object is running in multi-level secure partition 3, which has both *secret* and *top-secret* clearances. The database object has two methods, Secret_Read() and TopSecret_Read(), which require the invoking client to have secret and top-secret clearances, respectively.

The MLS policy for our example system is that all processes can only communicate with processes of equal or higher security clearances. The GIOP transport policy extends the MMR's MLS policy by placing further constraints on the GIOP messages sent between partitions the MMR already allows to communicate.

As Figure 3 shows, the MMR allows client A to communicate with the database object because partitions 1 and 3 both have secret clearance. The GIOP guard, however, restricts client A's communication with the database object to only invocations of the Secret_Read() method since A does not have the top-secret clearance required to invoke TopSecret_Read(). Client B is not allowed to access the database object at all. The MMR blocks all requests sent by B because partition 2 does not have the equivalent clearance(s).

Conclusion

The MILS architecture provides a cost-effective and efficient way to build verifiable secure systems. By separating security functionality into modular components, high-assurance systems can be engineered and evaluated much more rapidly and independently. The MILS architecture is an approach to system design that is supported by industry and government. SKs, the lowest layer of the MILS architecture, are already being deployed by multiple real-time operating system vendors [14]. Common criteria protection profiles are currently being developed for both the separation kernel [15] and MILS middleware [16].

In this article, we have shown how a security policy can be enforced on GIOP messages sent between MILS partitions. A policy that allows or disallows method invocations based upon the requesting client partition, the servant object, and the object method also can be used. Our test-bed implementations also show how guards can allow a MILS system to enforce both MLS and application-specific policies, thus providing fine-grained access control of inter-partition communication.◆

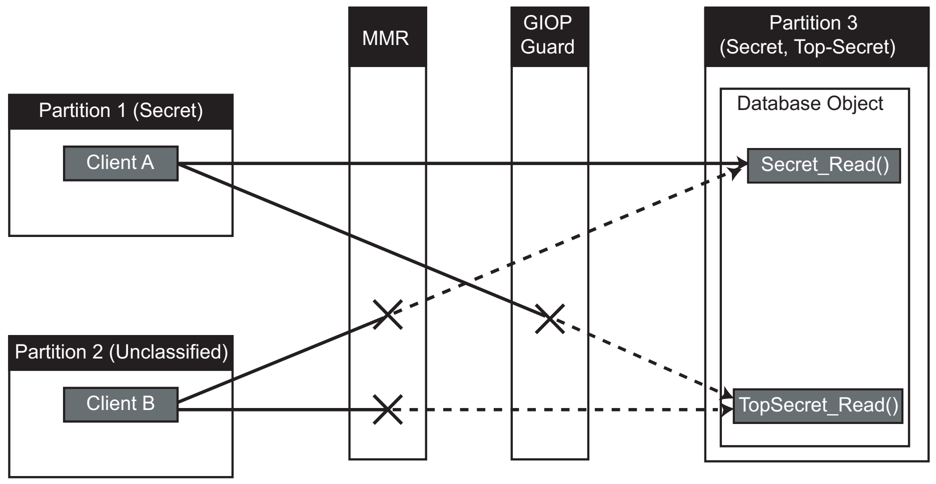
Clarification

This material is based on research sponsored by the AFRL and Defense Advanced Research Projects Agency (DARPA) under agreement number F30602-02-1-0178. The U.S. government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL and DARPA or the U.S. government.

References

1. Alves-Foss, Jim, W. Scott Harrison, Paul Oman, and Carol Taylor. "The MILS Architecture for High Assurance Embedded Systems." International Journal of Embedded Systems 2005 (to appear).
2. Radio Technical Commission for Aeronautics. "Software Considerations in Airborne Systems and Equipment Certification (RTCA DO-178B)." Washington: RTCA Inc., 1992 <www.rtca.org>.
3. Radio Technical Commission for Aeronautics. "Requirements Specification for Avionics Computer Resource (ACR) (RTCA DO-255)." Washington: RTCS Inc., 2000 <www.rtca.org>.
4. Common Criteria. CC Recognition Arrangement: Common Criteria for Information Technology Security Evaluation (Vers. 2.1). 2004 <www.commoncriteriaportal.org>.
5. Dransfield, Michael, et al. "MILS/MLS Architecture for Deeply Embedded Systems." NetCentric Operations 2004.
6. MacLaren, Lee. "New Options in Embedded Computing Security." Boeing Technical Excellence Conference, 2003.
7. Alves-Foss, Jim, Carol Taylor, and Paul Oman. A Multi-Layered Approach to Security in High Assurance Systems. Proc. of the Hawaii International Conference on System Sciences, 2004.
8. Adams, Charlotte. "Keeping Secrets in Integrated Avionics." Aviation Today Mar. 2004 <www.ghs.com/download/articles/Aviation_today.pdf>.
9. Miller, Alyson, Mark Jefferson, and Jeff Rogers. "Global Information Grid Architecture." The Edge: MITRE Advanced Technology Newsletter July 2001 <www.mitre.org/news/the_edge/july_01/miller.html>.
10. Ackermann, Robert. "Jointness Defines Priorities for the Defense Department's Global Grid." Signal Magazine Apr. 2001 <www.afcea.org/signal/articles/anmviewer.asp?a=120&z=115>.
11. Hanebutte, Nadine, et al. Software Mediators for Transparent Channel Control in Unbounded Environments. Proc. of the 6th IEEE Information Assurance Workshop, 2005.
12. Bell, D. Elliot, and Leonard LaPadula. "Secure Computer System: Unified Exposition and Multix Interpretation." ESD-TR-75-306. Bedford, MA: MITRE Corp., 1976.
13. Jajodia, Sushil, and Ravi Sahndu.

Figure 3: *An Example Policy*



Polyinstantiation Integrity in Multilevel Relations. Proc of the IEEE Symposium on Research into Security and Privacy, 1990 <http://citeseer.ist.psu.edu/121576.html>.
14 Ames, Ben. "Real-Time Software Goes Modular." Military and Aerospace Electronics. 14.9 (2003):24 <www.ghs.com/download/articles/GHS_RTOS_modular_090103.pdf>.
15. National Security Agency. U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness (Vers. 0.621). Washington: NSA, 2004.
16. University of Idaho. MILS CORBA Protection Profile, Vers. 0.52 (draft). Moscow, ID: Univ. of Idaho, 2003.

About the Authors

W. Scott Harrison, Ph.D., is an assistant professor in the Computer Science Department at the University of Idaho, where he has been since 1999. His current research involves information assurance and computer security issues. Harrison has a doctorate in computer science from Tulane University in New Orleans.

Center for Secure and Dependable Systems
University of Idaho
Moscow, ID 84844-1008
Phone: (208) 885-4114
Fax: (208) 885-7099
E-mail: harrison@cs.uidaho.edu

Paul W. Oman, Ph.D., is a professor of computer science at the University of Idaho. He held the distinction of University of Idaho Hewlett-Packard Engineering Chair for a period of seven years. Oman is a senior member in the Institute of Electrical and Electronics Engineers (IEEE) and is active in both the IEEE and the IEEE Computer Society. He has published more than 100 papers and technical reports on computer security, computer science education, and software engineering. He is a past assistant editor of *IEEE Computer* and *IEEE Software*.

Center for Secure and Dependable Systems
University of Idaho
Moscow, ID 84844-1008
Phone: (208) 885-4114
Fax: (208) 885-7099
E-mail: oman@cs.uidaho.edu

Nadine Hanebutte, Ph.D., is a postdoctoral fellow at the University of Idaho's Center for Dependable and Secure Systems. Her industrial experience includes work at the HypoVereinsbank in Munich, Germany as a software engineer for the risk-control department. Her current research includes software security, information assurance, and computer security education. Hanebutte has a Master of Science in computer science from Otto von Guericke University in Magdeburg, Germany, and a doctorate from the University of Idaho.

Center for Secure and Dependable Systems
University of Idaho
Moscow, ID 84844-1008
Phone: (208) 885-4114
Fax: (208) 885-7099
E-mail: hane@cs.uidaho.edu

Jim Alves-Foss, Ph.D., is an associate professor of computer science at the University of Idaho. He also serves as the director of the University of Idaho Center for Secure and Dependable Systems, which focuses on information assurance education and research. His current research involves the use of formal methods for protocol analysis and security policy verification.

Center for Secure and Dependable Systems
University of Idaho
Moscow, ID 84844-1008
Phone: (208) 885-4114
Fax: (208) 885-7099
E-mail: jimaf@cs.uidaho.edu

Letters to the Editor

Dear CrossTalk Editor,

It is often said today that the difference between software reliability and hardware reliability is that software does not fatigue, wear out, or burn out.

Thirty years ago, it was common to read of computers that would become sluggish and ineffective after running for a while. The simple treatment was to shut down and restart. Eventually it was realized that the problem was caused by poor memory management. The invention of *garbage disposal*, when used, takes care of it. Unfortunately even nowadays good memory management is not found as often as it should be, and Professor Trivedi at Duke University has done a deal of work trying to educate people in the effects of what he calls *aging* and what to do about it.

It could as easily be called *wear*. Consider the following: we speak of brakes wearing when they lose material, of structures wearing by fatigue (change in the crystalline structure), and of lubricating oil wearing out (chemical change). There is no common factor except simply reduced usability as a result of use. And software with badly managed memory suffers reduced usability as a result of use.

It is not a matter of which heated argument is worthwhile; however, unless software designers recognize that they must design to prevent reduced usability as a result of use, there will be systems that have to be frequently restarted, causing a nuisance.

Roderick Rees
Boeing

Dear CrossTalk Editor,

In her article, "How and Why to Use the Unified Modeling Language" (June 2005 CrossTalk), Lynn Sanderfer gave a useful survey of the UML and the benefits it can bring to a development process. It is also useful to discuss some of the disadvantages of UML.

The developers must learn from the users enough to make a domain description that is sufficient to create the application. UML is not a satisfactory notation for a domain description, because it is too hard for users to read.

There is no standard for saving and exchanging UML, so there is a risk to maintainability in being locked in to a proprietary tool. XML records the model but not the layout of the diagram, and the XML standard is a long way short of guaranteeing portability. In fact, there is a commercial market in XML conversion from one proprietary dialect to another.

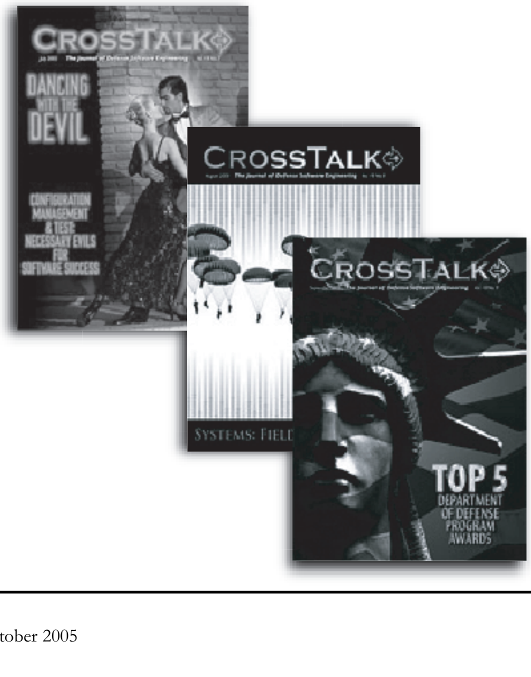
Most tools can export to the XML MetaData Interchange (XMI) format. Unfortunately XMI records the model but not the layout of the diagram, and the XMI standard is a long way short of guaranteeing portability. Not only are its versions very different, but proprietary extensions are common, and the Object Management Group does not provide tests for compliance. (In general, these are the features of UML that cannot be mapped to the Meta-Object Facility.)

Generating code from UML is not easy. There is no single tool or interface that has become standard for doing this. UML is stored in a binary file, so changes to the model by different developers cannot be merged automatically. The collaboration of a team of developers, especially a distributed team, depends on a source code repository and is founded on the diff utility. The use of UML leads to a directive style and a waterfall process, which is not suitable for all projects.

UML provides a good way to visualize object-oriented software. It is a suitable tool for some tasks. But a process in which a UML model is the central artifact is unsuitable for many projects.

Chris Morris
Daresbury Lab
Warrington, UK

Call for Articles



If your experience or research has produced information that could be useful to others, CrossTalk can get the word out. We are specifically looking for articles on software-related topics to supplement upcoming theme issues. Below is the submittal schedule for three areas of emphasis we are looking for:

Beyond Capability Maturity Model®
April 2006
Submission Deadline: November 21, 2005

Why Programs Fail
June 2006
Submission Deadline: January 16, 2006

Netcentricity
July 2006
Submission Deadline: February 13, 2006

Please follow the Author Guidelines for CrossTalk, available on the Internet at <www.stsc.hill.af.mil/crosstalk>. We accept article submissions on all software-related topics at any time, along with Letters to the Editor and BackTalk.

Application Security: Protecting the Soft Chewy Center

Alec Main
Cloakware

The software at the heart of many military systems is traditionally defended using the network security model in which vulnerable processing and information are encircled in defensive technologies. Unfortunately, network security is proving insufficient to defend against a wide range of attack scenarios targeting commonly available software. Likewise, system-level strategies are proving just as insufficient to defend against such attacks. However, effective application security provides protection from the inside out by implementing defensive techniques into applications and data themselves. As more commercial off-the-shelf hardware and software is used for military purposes, application security becomes more important since the skills and vulnerabilities that can be leveraged in attacks are more widely known. This article applies commercial lessons learned to military scenarios.

Software is at the core of all our military systems that provide technological advantage and strategic superiority over adversaries. Not only is software essential in delivering technological advantage, it also represents a significant portion of the defense program's operational capacity – and a significant investment.

When software applications fall into the hands of adversaries, they can analyze weapons, tactics, techniques, and procedures for vulnerabilities, and can produce countermeasures and more lethal weapons while saving research and development costs [1]. Military forces must prevent their information and weapons assets from being turned against them.

In the past, systems built with proprietary hardware and custom software were inherently more difficult to attack or reverse engineer. This approach, however, has become less desirable as commercial off-the-shelf (COTS) hardware, operating systems, and applications provide richer features, performance, and flexibility while reducing costs and deployment time.

Modern COTS software is complex, with a Windows or Unix operating system and major applications consisting of over 100 million lines of code. At an estimated frequency of one security bug per thousand lines of source code, a typical system will have over 100,000 security vulnerabilities [2]. By their nature, COTS systems are also widely available to and well understood by hackers. Information about vulnerabilities is widely shared, and tools for reverse engineering are readily available on the Internet. Military adoption of COTS software makes it potentially easier for foreign adversaries to reverse engineer and steal the technological advantage embodied in these software applications.

These facts are recognized by the Department of Defense (DoD). In December 2001, the Software Protection Initiative (SPI) was established to prevent the exploitation of national security application software by U.S. adversaries. As a U.S.-led initiative, the SPI is on the leading edge of determining the requirements for application security and guiding development of protection techniques. In addition to the two traditional components of information assurance – network security and operating system integrity – the SPI recognizes that an application-centric approach to protecting important DoD software is an essential *third leg* to the information assurance triad [3].

Inherent application security is also vital for critical infrastructure protection at home. Energy and utilities, communications, financial networks, transportation, and emergency and government services all depend on maintaining our applications and networks, which depend on software that currently lacks this third element in the triad of information assurance.

The DoD is acting to protect its software technology from reverse-engineering, unauthorized use, theft, and other types of exploitation. This article addresses modern techniques to make software applications inherently secure, and explores ways to protect vulnerable applications and the data they handle, including real-world examples.

The Need for Software Protection

Commercial and military network security specialists are beginning to recognize that application protection is a vital part of their overall security strategy. Perimeter security, in which we attempt to protect vulnerable assets in a trusted environment behind a secure perimeter, no longer works.

Firewalls were the first form of information technology perimeter defense, but they remain vulnerable to viruses and worms that penetrate and compromise internal systems using authorized network access vectors. Additional defenses such as virus scanners and intrusion detection systems are reactive and remain vulnerable to new threats. As networks expand, just defining the perimeter is challenging. Furthermore, perimeter defenses cannot protect application software from exploitation by insiders. With perimeter defenses breaking down, the result is software being deployed in inherently hostile environments.

Cryptographic technologies have long been used to protect data in transit and in storage. However, cryptography assumes that the end points, or points of use, are trusted. This is no longer the case as perimeter defenses such as firewalls break down and insiders can no longer be trusted. Compromised end-point systems can render even the best cryptography useless. Vendors of applications that present intellectual property in the form of music, movies, and games are representative of the realization that valuable encrypted data can only be fully protected by end-to-end security that encompasses the application software in addition to data encryption.

Application vulnerabilities are important for the military because they exist in one form or another in many military scenarios. History tells us that insider threats can never be ignored when designing military systems. COTS-based software compounds the threat by providing hostile parties with well-understood targets. But further, military software systems deployed in the field risk physical capture by adversaries.

Software protection [4, 5] is vital to defending these assets.

How Do We Protect the End-Points?

Currently, the common approach to protecting the end-points is to use system-level security by, for example, signing and/or encrypting all the software and data with a cryptographically strong key. But this approach looks remarkably like the old approach to network security – a perimeter or fortress defense – with a soft, chewy, vulnerable center.

The goal of system-level security is to build a *chain of trust* with the root of trust in hardware [6]. In this approach, the hardware validates the boot loader, which validates the kernel, which in turn validates the applications. Cryptographic [7] techniques are used to sign the code involved throughout the trust chain.

Code signing has its limitations. It is commonly used on the Internet to warn users of malicious software, called malware, and/or Trojan Horses that they might download. To be effective, the host must be trusted, every application must be signed, and the user must not want or – preferably – not even have the option to download unsigned software. However, the checking mechanism is typically not secure itself, meaning that code signing offers no protection when the host itself is untrusted or under attack.

Chain of Trust or House of Cards?

System security can be effective to a point; however, once broken, everything above the break, including the intellectual property and critical processing within the software applications, is exposed in much the same way as network security is breached once the firewall is penetrated.

System security is expensive to design and deploy in the first place and, once broken, is expensive and/or impractical to effectively and securely replace. Legacy issues also complicate upgrades needed to keep ahead of the latest hacking techniques. For these reasons, software-based renewable systems are being deployed by telephone companies for new, always-connected applications such as Internet Protocol Television (IPTV), where upgrades can be designed in, allowing controlled security updates to be pushed to the system across the network. Similarly, next-generation cable television security is moving to renewable software. Software-renewable systems through methods such as proactive obfuscation [8] are one means by which the military can deploy security that is both robust and affordable.

Example: Xbox

Microsoft's Xbox game console is an interesting public example of an intense reverse engineering and hacking effort [9]. Microsoft had two goals in locking down the Xbox: to prevent illegal copying of their games, and to prevent subsidy fraud by which a user could use the heavily subsidized hardware for unauthorized applications or purposes. The Xbox was designed to ensure that only signed applications could run, that only the original Microsoft approved code could be used on the Xbox, and that copied DVDs could not be used.

Microsoft realized that not only did the DVD authentication code need to be protected against reverse engineering, but the chain of trust needed to extend down to the boot loader to ensure that only a valid operating system could be loaded. Microsoft left the original boot loader in the ROM as a decoy that would entice attackers to waste their time. The true boot loader was then placed in the graphic controller and written to be self-verifying. The true boot loader contained the obfuscated key used to decrypt and load the kernel into memory. The kernel, in turn, verified that only signed applications were loaded and only original game DVDs were being used.

The huge popularity and ready availability of the Xbox and copy-protected games provided a large target for the hacker community. It took six months to finally break the protection. When the real boot loader was finally discovered and reverse engineered, the chain of trust came tumbling down. Copied DVDs and other applications could be run on the hardware. However, to replace the boot loader required a special *mod* chip that required cracking the Xbox cover and voiding the warranty.

Nevertheless, a second attack was then developed that exploited a buffer overflow, which meant no hardware modifications were necessary. Since code verification only occurred at boot-up, arbitrary code could be run independent of the system-level security.

The Xbox reminds us what happens when the chain of trust is broken and the house of cards comes tumbling down. The Xbox presented a challenging situation because the system was designed to be self-contained and autonomous – like many weapons systems or autonomous vehicles – and the user untrusted. Captured vehicles or undetonated bombs present a very similar type of threat, albeit infinitely more dangerous in the military arena.

But Will the Military Do It Differently?

Military systems can put more constraints on the user than a commercial producer like Microsoft can. One obvious improvement would be to include a hardware root of trust, either in a security chip such as a Trusted Platform Module, or a removable component like a smart card or dongle. The theory behind this approach is that the smart card can be removed or even physically destroyed by a trusted user prior to capture of the system.

The system must be designed to ensure all the appropriate software is encrypted and that a necessary secret lies in the removable key. Without the key stored on the smart card, the system cannot decrypt and run software applications. The system must still boot sufficiently to read the card, but if it is not present, an attacker will not be able to jeopardize the system.

But even this added level of protection has limitations that expose the military to potential security breaches. For example, the system could still be reverse engineered by a nation that legitimately purchased it and has access to the entire system. Without further precautions, the technology's soft, chewy center is exposed.

The same vulnerabilities would be exposed if the system were captured by an adversary when still running, or if the smart card were not removed, with serious consequences of disclosure and a possible weakening of other deployed systems.

A prime example is the U.S. Navy EP-3E Aries II surveillance plane that made an emergency landing on Hainan Island in China in April 2001 after a collision with a Chinese fighter jet. The plane was valued at U.S. $80 million and was packed with sophisticated eavesdropping equipment [10]. Military experts were concerned that Japan and the United States would have to change their secret communication system, at a cost of millions of dollars, as a result of Chinese scrutiny of the top-secret equipment [11]. Application security would have seriously hampered efforts to reverse engineer sensitive systems on board.

Missiles and bombs that do not explode are also vulnerable to reverse-engineering of weapons and guidance systems by adversaries. The United Nations mine clearing officials stated that between 10 percent and 30 percent of the missiles and bombs dropped on Afghanistan have not exploded [12]. According to a DoD briefing, a total of seven Tomahawk cruise missiles went astray in the Saudi Desert without exploding [13]. In Kosovo, Yugoslavia, evidence of missiles and bombs that had not exploded included a U.S. $1.25 million High Speed Anti Radar Missile on a highway and a Maverick antitank missile that had apparently missed its desired target and embedded itself in the roadside verge [14].

Planning further defenses against these scenarios without application security exposes the difficulty of trying to patch together highly specific defenses against every eventuality. Would a special launch sequence be required for missiles, loading a decryption key into memory from a removable smart card before launch? How would field officers know if an autonomous vehicle were downed and how would they take action to defend the system? Would sensors track its state so that random access memory could be cleared and the system shut down? These approaches each bring operational scenarios and key management issues into a battlefield environment with associated training, readiness, and the need for correct execution. As keys proliferate, the insider threat only further increases.

Application Security Provides Defense in Depth

To be truly effective, there must not be a soft chewy center, but hardened applications that are an active part of the chain of trust. A chain that establishes real trust is a chain mesh rather than a series of links. The hardware authenticates the loader, which authenticates the kernel, which authenticates the application. The application in turn authenticates the hardware, loader, and kernel while it is running, preventing one break at a low level from providing the keys to the castle. Application security provides protection against insiders even if system-level security is compromised. Reaction when tampering or an attack is detected can be as varied as logging, *calling home*, or an intentional destructive hostile response such as erasing hard drives or damaging central processing units.

Application security further increases the effort required to break system-level security because failure is not directly correlated to attacker actions. For example, tampering detected in one part of the software may result in subtle data errors that only become obvious elsewhere, making it difficult for a hacker to know where the attack was discovered. Unlike the perimeter tactics borrowed from network security, this provides a logical and complex form of defense in depth.

There are a number of application-level security techniques available to defend both source code and compiled applications. Typically, the best strategy is to combine multiple defenses in ways that are complementary and protect not only the application and data, but also each other.

Application Security Techniques

To understand application security techniques, it is best to understand the following traditional means by which software is attacked.

- **Analysis.** Classic reverse engineering and analysis of the software and protocols to identify vulnerabilities. This can be static analysis when the code is not running such as disassembly and decompilation, or dynamic tracing of the executing code using debuggers and emulators.
- **Tampering.** Modifying the code and/or data so that it performs according to the attacker's objectives rather than as designed.
- **Automation.** The creation of scripts or code to apply the tampering attack to multiple copies of the application. These are also known as class attacks or global breaks.

Application security techniques are used to prevent both static and dynamic analysis, as well as static and dynamic tampering to the software (see Table 1). This technology includes the following:

- **Code obfuscation.** Data flow and control flow transformations to prevent reverse engineering and tampering.
- **White-box cryptography** [15]**.** Specialized obfuscation techniques for cryptographic algorithms that prevent secret keys from appearing in memory during cryptographic operations.
- **Integrity verification.** Robust on-disk and in-memory verification to detect static tampering of the entire executable and dynamic tampering of code.
- **Anti-debug.** Detection and prevention of the use of debuggers and emulators to prevent dynamic analysis.
- **Code encryption.** Just-in-time decryption of code in memory to prevent static reverse engineering and tampering. Decompilers are ineffective with these techniques.
- **Diversity** [16]**.** A random seed is typically used by the tools that implement the software protection technology such that a different random seed creates a structurally different result. Diversity prevents automated or global attacks from being developed by adversaries against the protection techniques.

The first goal is to make static analysis difficult, time-consuming, and/or expensive. The obvious approach to prevent static analysis is to encrypt the binary. While there are techniques to extract these decrypted executables from memory, there are also source code techniques that prevent static analysis such as control flow flattening, which introduces pointer aliasing that can only be resolved at runtime. In addition, there are specific decompilation and disassembly prevention techniques that target these tools. Note that while very powerful disassembly tools exist, most low-level code written in C or C++ is very difficult to decompile with only a few tools available. Software protection is about using multiple layers of defense and all these techniques should be considered.

Runtime analysis of a system can be made very time-consuming by using anti-debugger and anti-emulation techniques. A range of techniques unto themselves, these can be effective on targeted platforms. The code can be tied to the platform via node locking and loading of new applications controlled by secure code signing techniques. Advanced just-in-time decryption (or self-modifying code) techniques also raise the bar against dynamic analysis. Authentication of components on the machine and encryption of communication channels, with protocol not subject to replay attacks, also prevent analysis. In addition, data flow transformation techniques can be used to hide and randomize data values even when operated on within main memory. White-box cryptography refers to specific cryptographic implementations designed to prevent key extraction even when the operation can be statically or dynamically viewed by an attacker. Steganographic techniques can also be used for key hiding.

Static tampering is prevented with binary encryption techniques, as well as by

Table 1: *Application Security Techniques and the Types of Attack They Defend Against*

	Reverse Engineering Attacks		Tampering Attacks		Automated Attacks
	Static	Dynamic	Static	Dynamic	
Code Obfuscation	✓	✓	✓	✓	Diversity
White-Box Cryptography	✓	✓			Diversity
Integrity Verification			✓	✓	Diversity
Anti-Debug		✓			Diversity
Code Encryption	✓		✓		Diversity

introducing data dependencies in the code to change an easy branch jamming attack into tampering – increasing the effort required and involving multiple changes to the code. An important technique to prevent tampering is code signing, but the code signing mechanism itself will be subject to attack and so must also be suitably hardened. Integrity verification of applications should be done statically (on-disk) as well as in memory to prevent dynamic tampering attacks.

Prevention of automated attacks is best achieved by deploying code and data diversity so that a successful attack will only work for a subset of users. Diversity of code is a result of most software protection techniques outlined above. It is similar to having different keys (diversity of data) for different users. Diversity of code recognizes that attacks will be made on the software in addition to the data. Automated attacks are also mitigated by software renewability, which can be made low cost – if designed in upfront. Conversely, with hardware-based security, renewability is a major cost. Software can be renewed selectively, proactively, or reactively – depending on the strategy and the attacks to the specific system.

Diversity is a prerequisite for successful renewability; otherwise, attackers will perform differential analysis. This is a powerful attack used to quickly determine the changes made to software upgrades and shorten the time to successfully hack.

These application security techniques are integral to the SPI. The SPI's goals include institutionalizing software protection as part of the application software life cycle and developing user-friendly protection techniques. Institutionalized and easy-to-use software protection techniques like those described above provide an additional layer of security that helps to ensure the availability of critical assets and infrastructure while maintaining the strategic lead in technologies critical to national defense.

Institutionalized application-level security techniques can help assure that the development environment is safe against insider threats. For example, obfuscation of source code and diversification of binaries during the development process can help prevent a production system from being reverse engineered despite being protected, because earlier prototypes were not. Every version sent to the field for testing can be protected distinctly from every other version at low cost. This sort of technique can be particularly important for the development of complex systems that involve a significant software engineering factory. Innovations developed by subgroups within the process can be hidden from participants elsewhere in the product development cycle.

Conclusion

When developers build security directly into their applications, they significantly strengthen resistance to attacks, even on open platforms and in hostile environments. While system-level security can be effective to a point, it does not address all the attack scenarios, especially the insider attack. Rather than a chain of trust, application-level security combines with network security and operating system integrity to form a chain mesh. COTS systems lower system development and deployment costs while application security maintains the protection needed in today's networked, technically advanced warfare. Combined, they offer superior value and flexibility in the quest for battle readiness and operational success.

References

1. Hughes, Jeff, and Dr. Martin R. Stytz. "Advancing Software Security – The Software Protection Initiative." 8th Annual International Command and Control Research and Technology Symposium, Wright Patterson Air Force Base, Ohio, 2003 <www.dodccrp.org/events/2003/8th_ICCRTS/pdf/137.pdf>.
2. Trusted Computing Group. "Enabling the Industry to Make Computing More Secure." Backgrounder Jan. 2005 <www.trustedcomputinggroup.org/downloads/background_docs/TCG_Backgrounder_revised_012605.pdf>.
3. Trusted Computing Group.
4. Main, A., and P.C. van Oorschot. "Software Protection and Application Security: Understanding the Battleground." International Course on State of the Art and Evolution of Computer Security and Industrial Cryptography. Heverlee, Belgium, June 2003 <www.scs.carleton.ca/%7Epaulv/papers/softprot8a.pdf>.
5. van Oorschot, P.C. Revisiting Software Protection. Proc. of 6th International Information Security Conference, 2003:1-13. Bristol, U.K., 2003. Springer-Verlag Lecture Notes in Computer Science, Oct. 2003: 2851.
6. Trusted Computing Group.
7. Menezes, A.J, P.C. van Oorschot, and S.A. Vanstone. Handbook of Applied Cryptography. 5th ed. CRC Press, 1996.
8. Schneider, F. B., L. Zhou. Distributed Trust: Supporting Fault Tolerance and Attack Tolerance. Jan. 2004.
9. Huang, Andrew. Hacking the Xbox: An Introduction to Reverse Engineering. No Starch Press, July 2003: 288.
10. Pan, Philip P. "U.S. Team Arrives in China to Examine Plane." The Washington Post 1 May 2001.
11. Chandler, Clay. "No Deal Reached On Plane: U.S., China Agree Only on More Talks." The Washington Post 20 Apr. 2001.
12. Ryan, Julie. "3,800 Civilians Killed: Unwelcome News of the 'War on Terror'." Dallas Peace Times Feb.-Mar. 2002 <www.dallaspeacecenter.org/dpt0202/civilian_deaths.htm>.
13. The Command Post. 29 Mar. 2003 <www.command-post.org/2_archives/2003_03_29. html>.
14. Beaver, Paul. "World: Europe Analysis: How Yugoslavia Hid its Tanks." BBC News 25 June 1999 <http://news.bbc.co.uk/1/hi/world/europe/377943.stm>.
15. Chow, S., P. Eisen, H. Johnson, and P.C. van Oorschot. White-Box Cryptography and an AES Implementation. Proc. of the 9th International Workshop on Selected Areas in Cryptography, 2002: 250-270. Springer-Verlag Lecture Notes in Computer Science 2003: 2595, 2003.
16. Anckaert, Bertrand, Bjorn De Sutter, and Koen De Bosschere. Software Piracy Prevention Through Diversity. Proc. of the 4th ACM Workshop on Digital Rights Management.

About the Author

Alec Main is chief technology officer at Cloakware. He has been involved in the design and implementation of numerous software security systems for PCs, mobile devices, and set-top boxes, and has pioneered Cloakware's software protection technology. Main is a regular speaker at conferences on the topic of software protection and has a degree in electrical engineering.

Cloakware
8320 Old Courthouse RD
STE 201
Vienna, VA 22182
Phone: (866) 465-4517
Fax: (703) 760-7899
E-mail: alec.main@cloakware.com

Web Sites

SecurityFocus
www.securityfocus.com
SecurityFocus is a comprehensive and trusted source of security information on the Internet. SecurityFocus provides objective, timely, and comprehensive security information to all members of the security community, from end-users and network administrators to security consultants, information technology managers, chief information officers, and chief security officers.

Home PC Firewall Guide
www.firewallguide.com/spyware.htm
This posting of an anti-spyware guide on Home PC Firewall Guide provides links to help you fight spyware. Spyware has a range of meanings including keystroke-loggers, malware, browser parasites, unsolicited commercial software, scumware, home page hijackers, dialers, and Trojan horses. Microsoft Windows and Internet Explorer are the targets for most of these attackers. Home PC Firewall Guide provides easy access to basic information, and independent, third-party reviews of Internet security and privacy products for home, telecommuter, and small office and/or home office end-users.

KRC Anti-Spyware Tutorial
www.greyknight17.com/spyware.htm
This tutorial walks you through the basics of understanding what type of spyware attack you may encounter to instructions for removing spyware and various software assistance tools to spyware prevention. Its step-by-step tutorial instructs you on how to install and use common free spyware removal and protection tools. It also includes numerous links to these tools and industry information and definitions to aid in understanding. The main Web site, Kevin's Resource Center, offers help with computer-related issues. It includes links to sites that have tech support, free software downloads, and other resources.

WatchGuard
www.watchguard.com/infocenter/editorial/15860.asp
This posting at WatchGuard is a good basic article on spyware remediation, including a list of anti-spyware tools and links to more in-depth information. The main Web site also features white papers and case studies for review.

US-CERT
www.us-cert.gov
The U.S. Computer Emergency Readiness Team (US-CERT®) is a partnership between the Department of Homeland Security and the public and private sectors. US-CERT was established in 2003 to protect the nation's Internet infrastructure by coordinating defense against and response to cyber attacks. US-CERT is responsible for the following:
- Analyzing and reducing cyber threats and vulnerabilities.
- Disseminating cyber threat warning information.
- Coordinating incident response activities.

US-CERT interacts with federal agencies, industry, the research community, state and local governments, and others to disseminate reasoned and actionable cyber security information to the public. Information is available from the US-CERT Web site and through mailing lists. US-CERT also provides a way for citizens, businesses, and other institutions to communicate and coordinate directly with the U.S. government about cyber security.

Department of Homeland Security
www.dhs.gov/dhspublic
The DHS is the government's 15th cabinet-level agency, consolidating 22 previously disparate agencies under one unified roof. The DHS has three primary missions: Prevent terrorist attacks within the United States, reduce America's vulnerability to terrorism, and minimize the damage from potential attacks and natural disasters. The department's first priority is to protect the nation against further terrorist attacks. Component agencies will analyze threats and intelligence, guard U.S. borders and airports, protect critical infrastructure, and coordinate the response of the nation for future emergencies. Besides providing a better-coordinated defense of the homeland, DHS is also dedicated to protecting the rights of American citizens and enhancing public services such as natural disaster assistance and citizenship services by dedicating offices to these important missions.

Software Technology Support Center
www.stsc.hill.af.mil
The Software Technology Support Center (STSC) is an Air Force organization established to help other U.S. government organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and to accurately predict the cost and schedule of their delivery. The STSC specializes in systems engineering and development, software quality and test, project management, cost estimation, and software acquisition management.

Institute of Electrical and Electronics Engineers
www.ieee.org
The Institute of Electrical and Electronics Engineers (IEEE) promotes the engineering process of creating, developing, integrating, sharing, and applying knowledge about electrical and information technologies and sciences. IEEE provides technical publications, conferences, career development assistance, financial services and more.

The Data & Analysis Center for Software
http://iac.dtic.mil/dacs/
The Data & Analysis Center for Software (DACS) is a Department of Defense (DoD) Information Analysis Center. The DACS is the DoD Software Information Clearinghouse for state-of-the-art software information providing technical support for the software community. The DACS offers a wide variety of technical services designed to support the development, testing, validation, and transitioning of software engineering technology.

Best Practices Clearinghouse
http://fc-md.umd.edu/bpch/
The Department of Defense (DoD) Best Practices Clearinghouse was established to improve DoD's acquisition of software-intensive systems by helping programs select and implement proven acquisition, development, and systems engineering practices appropriate to their individual programmatic needs. It will support component improvement initiatives by enabling acquisition organizations to create and institutionalize effective system acquisition processes and maintain well-trained, experienced personnel.

From: Hiram Rextall, TLA Security Head for Information Technology
To: All TLA Company Personnel
Subject: Network Passwords

As you know, restricting access to information is a cornerstone of corporate TLA culture. Unauthorized or unwanted access to the TLA company network by a person or persons unknown can have a consequence or consequences unknown. Your password is your passport to the world of corporate knowledge. Your password must be protected at all times, at all costs, and at all points of ingress or egress. Therefore, effective immediately, the following rules regarding passwords are in place and will be strictly enforced:

1. Passwords must be between 23 and 37 alphanumeric characters. (As 23 and 37 are prime numbers, their use as bounds should confuse decryption techniques.)
2. Passwords must contain at least three members from each of the following four groups: uppercase letters, lowercase letters, numbers, and special characters.
3. The password may not contain more than five consecutive members of any of the four groups listed in No. 2.
4. White space characters (space, tab, etc.) are allowed, provided a network administrator visually verifies them.
5. Because all computer data is ultimately stored in binary format (a combination of ones and zeros), the numbers 1 and 0 may not be used in a password.
6. Because the letter l looks like the number 1 and the letter O looks like the number 0, you cannot use l or O either. Similarly, you cannot use an exclamation point (!) or vertical bar (|).
7. Lowercase letter *o* may be used provided it is not next to a number.
8. The password may not contain any word in the English language, including *I* (uppercase only) and *a* (either case). (Company personnel operating abroad: This word restriction applies in the local language as well.)
9. The password may not contain your name, your employee number, your street address, your mother's maiden name, your pet's name, your first grade teacher, or your supposedly secret drag queen/biker momma name. (The sorts of people who try to break into networks know that last one.)
10. Passwords are to be changed every 24 hours. Because the IT department cares, you do not have to change your password every time you log in. However, should you log out without having changed your password and need to log in again within the same 24-hour period, you will have to submit a request to the IT department to have a temporary password issued. This will probably take 24 hours to process. Days are considered to begin at 0000 Zulu hours.
11. The password cycle is synchronized with each 24-hour cycle. If you fail to log in during a 24-hour period, it will be necessary to create and use passwords covering the missed period. For example, upon return from a two-week vacation it will necessary to update your password 14 times.
12. Passwords are to be memorized. They are not to be recorded using any media: paper, computer file, clay, wood, iron, sand, wax, Styrofoam, food, or animal parts.
13. Passwords may be reused after one year.
14. You may not use any other user's current or past passwords; the one-year rule applies here.

It may seem difficult to come up with passwords that meet these requirements and are mnemonically friendly. Please keep in mind that you only need to remember each password for a single day. It may help you to associate something familiar to assist your memory with your password. For example, "The quick brown fox jumped over the lazy dog" could be used to associate to the password Dz23+8uVC**ojy~xiMn4_Q?. (Note: This password has already been taken.)

While mathematicians are still crunching the numbers, it is calculated that there are over 10^{14} passwords possible. However, TLA management is concerned that we may run out of passwords. To address this, IT is already working on a three-part login system (TriHard) that will make passwords obsolete. The three TriHard components include:

- Biometric fingerprint scan
- Simultaneous spring-loaded needle prick for blood sample and DNA comparison
- Optical retinal scan

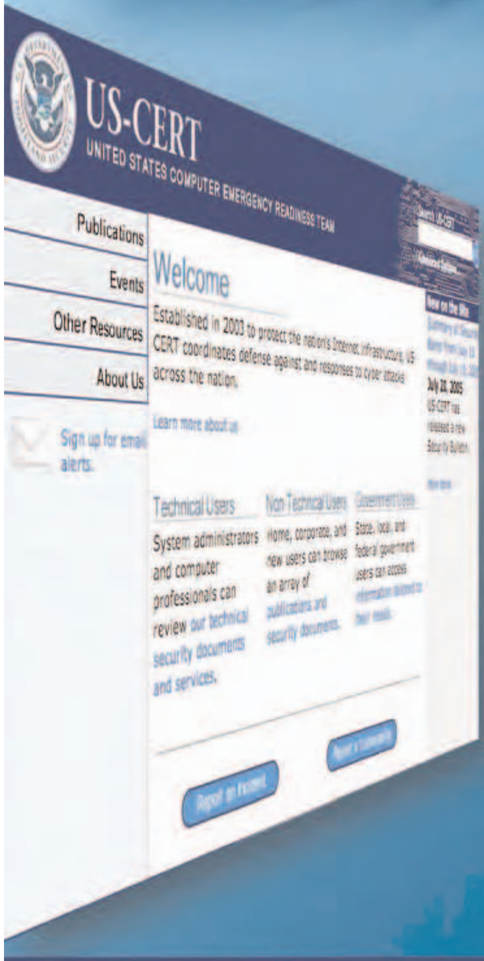
A TriHard prototype involving its last two components is in alpha testing. Volunteers have been hard to come by for some reason. IT personnel will be visiting your work areas soon seeking assistance. Keep an eye open for them.

"Remember, you can't do it without IT."
– Hiram Rextall

— Dan Knauer
TLA – A Three Letter Acronym Corporation

Homeland Security

Software Assurance Program


US-CERT
UNITED STATES COMPUTER EMERGENCY READINESS TEAM

Publications
Events
Other Resources
About Us
Sign up for email alerts.

Welcome

Established in 2003 to protect the nation's Internet infrastructure, US-CERT coordinates defense against and responses to cyber attacks across the nation.

Learn more about us

Technical Users	Non-Technical Users	Government Users
System administrators and computer professionals can review our technical security documents and services.	Home, corporate, and new users can browse an array of publications and security documents.	State, local, and federal government users can access information tailored to their needs.

Software is essential to enabling the nation's critical infrastructure. To ensure the integrity of that infrastructure, the software that controls and operates it must be reliable and secure.

Security must be "built in" and supported throughout the lifecycle.

Visit http://BuildSecurityIn.us-cert.gov to learn more about the practices for developing and delivering software to provide the requisite assurance.

Sign up to become a free subscriber and receive notices of updates.

http://BuildSecurityIn.us-cert.gov

The Department of Homeland Security provides the public-private framework for shifting the paradigm from "patch management" to "software assurance."

CrossTalk is co-sponsored by the following organizations:

Software Engineering Division
Ogden Air Logistics Center

Homeland Security

CrossTalk / 309 SMXG/MXDB
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737